# Reconnaissance de structures bidimensionnelles : Application aux expressions mathématiques manuscrites en-ligne

Ahmad-Montaser Awal

**UNIVERSITE DE NANTES**

**ÉCOLE DOCTORALE**

« SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE MATHEMATIQUES »

Année : 2010

## Thèse de Doctorat de l'Université de Nantes

Spécialité : Automatique et Informatique appliquée

*présentée et soutenue publiquement par*

**Ahmad Montaser AWAL**

*le 12 novembre 2010*

*à l'Ecole polytechnique de l'université de Nantes*

# Reconnaissance de structures bidimensionnelles : Application aux expressions mathématiques manuscrites en-ligne

Jury

| | | | |
|---|---|---|---|
| Président | : | M. Thierry PAQUET | Professeur, université de Rouen |
| Rapporteurs | : | M. Eric ANQUETIL | Professeur, INSA de Rennes |
| | | M. Mohamed CHERIET | Professeur, ETS, Montréal |
| Examinateurs | : | M. Pierre-Michel LALLICAN | Docteur, directeur RD, Vision Objects |
| | | M. Harold MOUCHERE | Maître de conférences, IUT Nantes |
| | | M. Christian VIARD-GAUDIN | Professeur, IUT Nantes |

**Directeur de Thèse : Christian VIARD-GAUDIN**
Laboratoire : IRCCyN
Co-encadrant : Harold MOUCHERE
Laboratoire : IRCCyN
Composante de rattachement du directeur de thèse : IUT de Nantes

N° ED 503 –105

**Remerciements**

Je tiens tout d'abord à remercier très sincèrement les membres du jury d'avoir accepté de m'accorder de leur temps précieux pour juger mes travaux de thèse. Je voudrais remercier en particulier Thierry Paquet de m'avoir fait l'honneur de présider ce jury. Je remercie également Eric Anquetil et Mohamed Cheriet d'avoir consacré un temps important pour rapporter sur le manuscrit de thèse et pour toutes les remarques approfondies sur le contenu scientifique de ce manuscrit.

Je souhaiterais remercier aussi l'entreprise Vision Objects pour son aide lors de la réalisation des parties techniques de ce travail. Je remercie en particulier Pierre Michel Lallican d'accepter d'être examinateur de ce travail mais aussi pour ses conseils précieux et tout l'aide qu'il m'a donnée.

Je ne remercierai jamais assez mon directeur de thèse Christian Viard-Gaudin et mon co-encadrant de thèse Harold Mouchère pour la grande qualité d'encadrement, pour leurs idées, pour toutes leurs aides, pour tout le temps qu'ils m'ont consacré et surtout leur patience. Merci à Christian qui m'a introduit au monde scientifique, m'a ouvert grand la porte sur le domaine de la reconnaissance des formes, et m'a appris comment diriger une recherche pertinente. Merci à Harold qui a lui aussi dynamisé ce travail et a enrichi les idées et les résultats de cette thèse.

Je remercie l'ensemble de l'équipe IVC pour son accueil et sa gentillesse sous la direction de Patrick Le Callet.

Je voudrais remercier Mazen Said et Ayman Hammade de l'université d'Alep qui m'ont donné l'occasion de continuer mes études en France. Je remercie également mes collègues d'études qui m'ont accompagné tout au long de mon parcours. Je remercie surtout mes amis qui m'ont soutenu pendant cette thèse.

Merci à ma famille, et notamment à mes parents, qui m'ont toujours soutenu dans mes études et ont fait de gros sacrifices afin que je puisse arriver à ce niveau. Merci également à mes sœurs qui m'ont toujours encouragé pour que mes rêves deviennent réalité.

Finalement, un très grand merci à Susan qui a toujours cru à mes compétences et m'a aidé énormément. Merci de m'avoir soutenu au quotidien et, d'avoir supporté mon manque de disponibilité.

# Table des matières

# Table des figures

# Liste des tableaux

# Liste des algorithmes

# INTRODUCTION GÉNÉRALE

Les premières traces de l'écriture remontent à environ 3300 ans avant J.-C., il y a donc un peu plus de 5000 ans, ce qui représente une période relativement courte à l'apparition de l'homme. Symboliquement, cette apparition de l'écriture marque le passage de la préhistoire à l'histoire et correspond dès lors à une mutation profonde et à une évolution rapide des sociétés humaines. Le développement progressif des civilisations a imposé le besoin de trouver un moyen pour conserver les lois, échanger et transmettre les idées et les résultats des travaux des savants de l'époque. Les archéologues ont trouvé des premières traces de conservation de l'écriture dans plusieurs régions de l'ancien monde, notamment : la Mésopotamie, le Levant, l'Egypte, ainsi que la Perse et la Grèce antique. Des tablettes d'argile et de pierre ont été utilisées initialement comme moyen de transmission de ces informations, avant que les Chinois inventent le papier au cours du second siècle. L'écriture manuscrite restait le seul moyen pour communiquer jusqu'à l'invention de l'imprimerie qui a permis à l'écriture typographiée de dominer jusqu'à nos jours. Un très intéressant dossier est consacré à l'aventure des écritures sur le site de la Bibliothèque Nationale de France, nous conseillons au lecteur curieux de s'y reporter[1].

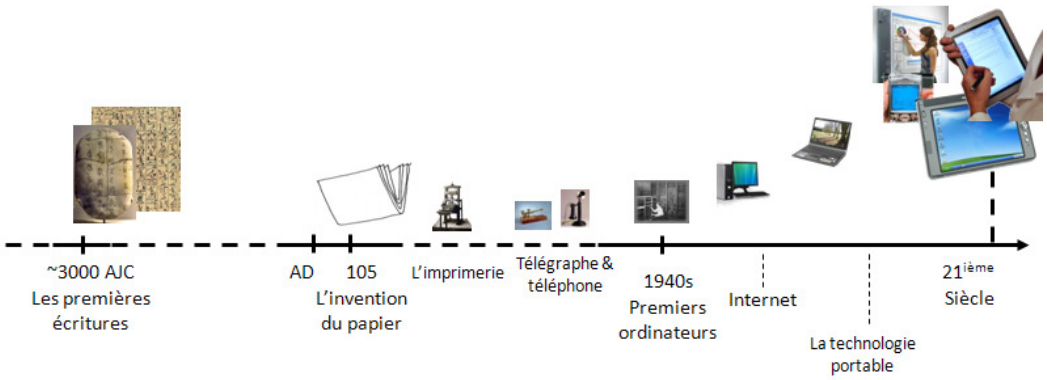



Figure 1 - L'évolution des moyens de communication

La communication manuscrite ne se limite pas à l'écriture de textes, mais va au delà si l'on considère les graphiques, dessins, schémas électriques, équations chimiques, plans d'ingénierie, … et de manière générale tout ce qui peut être représenté à main levée. L'écriture manuscrite est une compétence développée depuis la naissance qui permet à chaque individu de s'exprimer et de faire passer ses idées, à un tel degré que l'on peut dire que l'écriture de chacun est une empreinte distinctive et unique de cette personne.

Certaines personnes pensent que les progrès technologiques peuvent mettre l'écriture manuscrite en danger. En effet, on compte des millions de documents numériques (comportant du texte, des équations, des schémas, etc.) qui facilitent l'échange d'informations sur les réseaux (tel Internet) à travers des milliers des kilomètres en quelques secondes.

[1] http://classes.bnf.fr/dossiecr/

Cependant, l'écriture manuscrite reste une méthode privilégiée pour une grande majorité d'exprimer des idées et d'échanger des informations. Il en résulte l'invention de nouveaux outils pour faciliter l'intégration de l'écriture dans le monde numérique. C'est le cas par exemple des stylos digitaux, des tableaux blancs interactifs, des PDAs, des tablets PC et des téléphones mobiles avec écrans tactiles. Ces nouvelles technologies matérielles supposent en parallèle le développement de puissants logiciels permettant de profiter de cette évolution. Ainsi, les systèmes de reconnaissance de l'écriture ont fait leur apparition pour garantir la liberté pour l'utilisateur de saisir de l'information en utilisant sa propre écriture. En même temps, ils permettent la conversion de ces écritures sous forme numérique, ce qui ouvre la voie pour profiter de la puissance des traitements informatiques.

L'objectif des travaux de cette thèse est de proposer une contribution pour la reconnaissance des langages manuscrits bidimensionnels. Plus particulièrement, nous nous intéressons à la reconnaissance d'expressions mathématiques manuscrites en-ligne. La plupart des recherches émergeantes dans ce domaine considèrent un ensemble de sous-classes des expressions mathématiques et obtiennent alors des résultats prometteurs. Nombre de ces recherches abordent la reconnaissance d'expressions mathématiques en tant qu'une suite de sous tâches indépendantes. En conséquence, les erreurs se propagent d'une étape à l'autre.

Nous proposons dans cette thèse une architecture effectuant une optimisation simultanée de la segmentation, la reconnaissance et l'interprétation des expressions mathématiques. Cette optimisation se base sur une fonction de coût global. Dans ce cadre, le classifieur utilisé pour la reconnaissance de symboles est appris au sein du système global de reconnaissance d'expressions. Cet apprentissage global permet d'apprendre les symboles au cours de la segmentation au lieu d'utiliser un classifieur simplement appris sur une base de symboles isolés. Plus précisément, nous focalisons nos recherches sur les axes suivants :

- Améliorer la reconnaissance des symboles mathématiques au sein de l'architecture globale en proposant plusieurs stratégies d'apprentissage des classifieurs (isolé, global, hybride, isolé étendu en global).
- Proposer différentes méthodes d'analyse contextuelle et structurelle au sein de la structure globale. Nous avons choisi d'introduire une modélisation spatiale des relations entre les composantes d'une expression mathématique. Chaque règle de la grammaire est associée à une relation spatiale différente qui décrit la relation logique entre ses constituants. Les relations sont associées à des fonctions de coût qui pénalisent plus ou moins les hypothèses selon la position et la taille relatives de leurs constituants par rapport à une référence.
- Sélectionner des corpus et construire des bases d'expressions manuscrites en-ligne. En effet, il faut souligner l'indisponibilité de telles données. Nous

proposons d'utiliser un générateur d'expressions (LaTeX2Ink) traduisant une chaîne LaTeX quelconque en une expression manuscrite réaliste représentative de cette expression. Cet outil permet de facilement générer de grandes bases d'expressions mathématiques manuscrites en-ligne. Les avantages d'une telle approche sont multiples, mais bien entendu, on en voit également clairement ses limites, et cet outil ne dispensera pas à terme de collecter des expressions complètes pour réaliser une base de test comparative. Cette collecte réalisée récemment va permettre de mieux valoriser les différentes approches et méthodes proposées.

Les résultats obtenus sont très compétitifs et prometteurs. Ils ont donné lieu à plusieurs publications dans des conférences internationales (ICDAR'09, ICFHR'10, DRR'10, DRR'11) et nationales (Cifed'08, Cifed'10, RFIA'10).

Le reste de ce manuscrit est organisé comme suit :

Dans ***le chapitre 1*** nous introduisons la problématique générale de la reconnaissance de structures bidimensionnelles ainsi que le cadre et les motivations de nos travaux. Ensuite, nous mettons en lumière les difficultés associées particulièrement à la reconnaissance d'expressions mathématiques. Ce chapitre se termine avec quelques exemples d'applications potentielles des systèmes de reconnaissance de structures bidimensionnelles.

***Le chapitre 2*** se concentre sur l'état de l'art de ce domaine de recherche en explorant les méthodes et les systèmes proposés pour la reconnaissance de langages 2D en général. Nous nous focalisons plus sur ceux dédiés à la reconnaissance d'expressions mathématiques. Le chapitre est divisé en trois parties principales : la segmentation, la reconnaissance des symboles, et l'interprétation. Nous introduisons à la fin un résumé du système proposé lors des travaux de thèse de Guihuan Feng [**1**] qui a servi de point de départ à nos travaux.

Le problème de l'évaluation des systèmes de reconnaissance d'expressions mathématiques est détaillé dans ***le chapitre 3***. L'outil LaTeX2Ink est également présenté dans ce chapitre. Les bases d'expressions synthétiques (générées) ainsi que des bases récoltées sont introduites. Finalement, nous présentons les mesures les plus classiques pour l'évaluation et celles que nous utilisons pour évaluer nos travaux.

***Le chapitre 4*** se focalise sur la présentation de l'architecture globale de notre système. Tout d'abord, nous introduisons les classifieurs de symboles et la méthodologie d'apprentissage globale pour entraîner ces classifieurs. Nous proposons plusieurs variantes de classifieurs pour qu'ils soient plus adaptés au problème de reconnaissance d'expressions mathématiques. Ensuite, nous développons plusieurs méthodes pour l'analyse structurelle d'une expression mathématique.

***Le chapitre 5*** est dédié aux expérimentations réalisées et à la présentation des résultats afin de valider les différentes méthodes proposées. Nous présentons

également dans ce chapitre deux démonstrateurs basés sur notre système de reconnaissance d'expressions mathématiques.

Finalement, nous conclurons nos travaux et présenterons aussi quelques perspectives connexes à ceux-ci.

# CHAPITRE 1 : RECONNAISSANCE DE STRUCTURES BIDIMENSIONNELLES

## 1.1. Contexte

L'émergence des nouvelles technologies, telles que stylos digitaux avec processeurs embarqués, tablettes, smartphones, tableaux blancs interactifs, etc. a imposé de nouvelles formes de communication écrite manuscrite. Les travaux menés au cours de cette thèse font partie du projet ANR « Conversion et Indexation de l'Ecriture en Ligne » (CIEL) qui s'intéresse à ces nouvelles formes de communication. L'objectif du projet est de proposer des modèles, méthodes et systèmes de traitements pour appréhender un nouveau média : les documents manuscrits en-ligne. Il cherche à combler le fossé entre les documents électroniques et les documents papiers pour apporter à ceux-ci des fonctionnalités de haut niveau en profitant de la puissance de traitements automatiques.

Plus particulièrement, il faut disposer d'outils capables de transformer l'écriture de sa forme manuscrite en une forme numérique compréhensible par l'ordinateur. La nature aléatoire de l'écriture manuscrite, et la grande variété des styles de l'écriture rendent la tâche très difficile. Cette transformation de la forme manuscrite (physique) à une forme numérique (logique) est baptisée « Reconnaissance » de l'écriture. Il faut noter que les systèmes de reconnaissance de l'écriture ont connu un essor important ces dernières années [**2**] avec des améliorations très significatives des performances. Ces progrès ont été obtenus grâce à la convergence de multiples facteurs : augmentation de la puissance des calculateurs, disponibilité de très grosses bases de données pour l'apprentissage, utilisation de nouvelles techniques d'apprentissage. En conséquence, des applications ciblées pour des problèmes spécifiques tels que la lecture automatique des chèques, la lecture d'adresse de courriers postaux, etc. sont maintenant largement utilisées avec des niveaux de performances très élevés. Ces systèmes prennent avantage de la connaissance préalable de ce que l'on attend à reconnaître. Ainsi, pour la reconnaissance du montant littéral sur les chèques, le lexique est fermé et limité à quelques dizaines de mots. D'une manière générale, plus le contexte de la reconnaissance est maîtrisé, meilleures seront les performances de reconnaissance. Dans cet esprit, si la taille du lexique devient importante, typiquement de quelques dizaines de milliers de mots à plusieurs centaines de milliers de mots, il est alors indispensable d'accompagner le système de reconnaissance d'un véritable modèle de langage qui permettra de désambiguïser des situations visuellement complexes.

Toutefois, ces systèmes se limitent à reconnaître des textes qui sont constitués d'une séquence de caractères organisés eux-mêmes en séquences de mots appartenant à un certain vocabulaire. Dans ces systèmes, la disposition spatiale des constituants est ainsi réductible à une seule dimension, la ligne horizontale (ou verticale dans certaines langues asiatiques). Il n'en est plus de même lorsque la grammaire qui contrôle le langage produit est de nature

bidimensionnelle. C'est le cas en particulier pour les expressions mathématiques [**3**], les schémas [**1**], les diagrammes [**4**] [**5**], les tableaux [**6**], les équations chimiques [**7**], les partitions musicales [**8**], les caractères de certaines langues comme le Chinois [**9**], … et tout ce qui se dessine dans un espace bidimensionnel. Dès lors, de nouvelles approches permettant d'analyser ces objets présents dans de nombreux documents manuscrits doivent être envisagés. Il s'agit, là aussi, de résoudre les problèmes de segmentation, de reconnaissance et d'interprétation, mais cette fois dans un contexte bidimensionnel (2D).

## 1.2. Caractéristiques de l'écriture manuscrite

En général, la reconnaissance optique de caractères (OCR : Optical Character Recognition) implique de concevoir des systèmes capables de transformer des textes typographiés ou manuscrits dans un format compréhensible par l'ordinateur. Initialement, l'OCR était dédiée à la reconnaissance de caractères, mais grâce aux progrès technologiques, ce domaine a connu beaucoup de développement. Des systèmes de reconnaissance de mots, textes, documents, schémas, etc. sont largement étudiés par les communautés de reconnaissance de formes et de vision par ordinateur. Néanmoins par abus de langage et pour simplifier, nous appelons tout ce qui est écrit ou dessiné à la main : un signal manuscrit, que ce soit du texte, un diagramme ou même une expression mathématique. Comme on le voit sur la Figure 2, l'étude de la reconnaissance de l'écriture manuscrite se divise en deux domaines : hors-ligne et en-ligne selon la source numérique du signal de l'écriture. Dans le cas d'un signal en-ligne, deux stratégies de reconnaissance peuvent être utilisées : l'interprétation à la volée ou l'interprétation a posteriori.

Des avancées significatives ont été réalisées dans le domaine de la reconnaissance de l'écriture manuscrite depuis une trentaine d'années [**2**] [**10**]. Cependant, il reste toujours des problèmes et des difficultés qui ne sont pas complètement surmontés. On essaye alors de résoudre ces problèmes en imposant des contraintes d'entrée mais aussi en impliquant des connaissances du domaine en question.

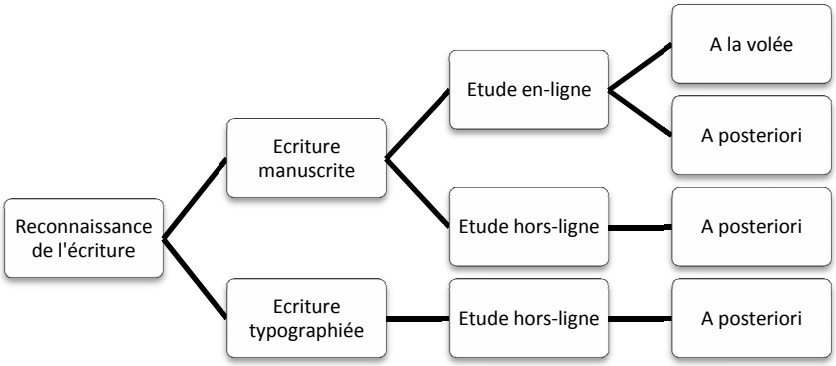



Figure 2 - Les domaines des systèmes de reconnaissance selon la nature de l'écriture et leurs stratégies de reconnaissance

***Hors-ligne vs en-ligne***

Un signal hors-ligne est le résultat d'une numérisation d'une écriture manuscrite déjà présente sur un support généralement papier. La numérisation peut être effectuée par un scanneur, un appareil photo, ou encore en extrayant des informations hors-ligne d'une entrée en-ligne. Le signal est donc disponible sous forme d'image. Avant d'aborder la reconnaissance proprement dite de l'écriture, il faut d'abord nettoyer l'image car elle est souvent bruitée à cause de la numérisation. Plusieurs étapes de traitement d'images sont nécessaires, comme par exemple : la binarisation, les opérations morphologiques, la squelettisation, …etc. Toutes ces étapes préliminaires rendent les systèmes de reconnaissance de l'écriture hors-ligne très lourds et les destinent plus particulièrement aux applications industrielles. Ainsi, des applications de reconnaissance de l'écriture hors-ligne sont utilisées à grande échelle pour effectuer des tâches très précises comme la lecture automatique de chèques, la lecture des adresses postales, la numérisation des documents d'archives ou encore le traitement des formulaires.

Comme le montre la Figure 3, un signal d'écriture hors-ligne se présente sous forme d'image. Ainsi, c'est une matrice d'éléments en deux dimensions dont chaque élément (pixel) est soit noir ou blanc, en niveaux de gris ou en couleurs. On ne possède par contre aucune information sur la dynamique de l'écriture elle-même. C'est pourquoi, le signal d'écriture hors-ligne est qualifié de ***statique***, il nécessite un traitement supplémentaire pour obtenir des informations sur la chronologie du signal [**11**] [**12**] [**13**].

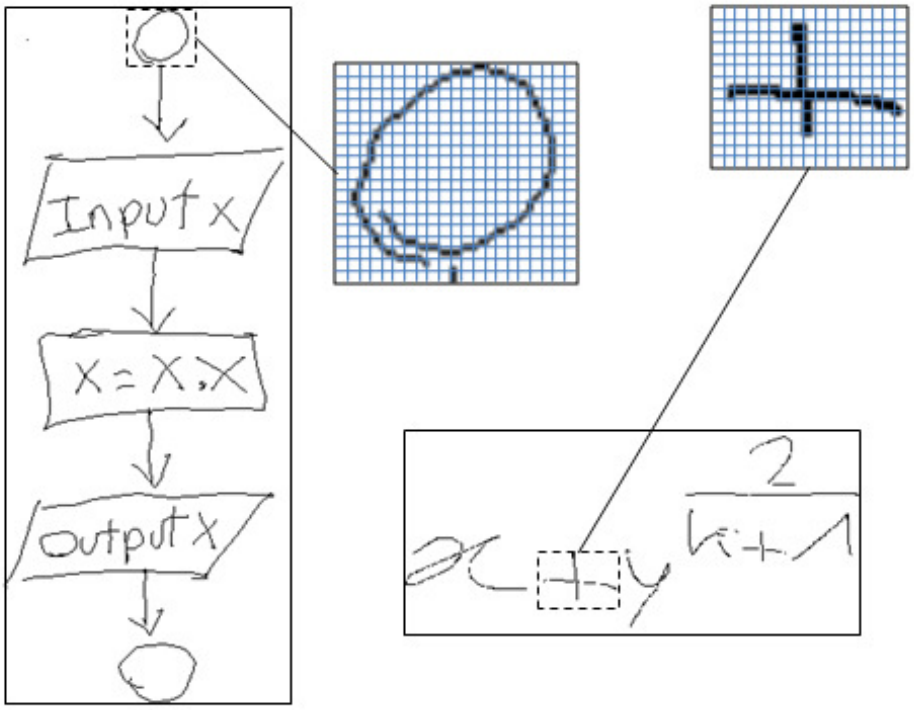



Figure 3 - Exemple du signal hors-ligne d'un diagramme et d'une expression mathématique

De façon complémentaire, un signal d'écriture en-ligne s'obtient grâce aux technologies permettant de numériser l'écriture au moment de sa production. La trajectoire de l'instrument d'écriture (doigt, stylet, stylo) est enregistrée lorsque celui-ci est posé sur la surface d'écriture (écran tactile, tableau, papier). On obtient donc les coordonnées de la trajectoire échantillonnée, généralement à intervalle constant de temps (typiquement toutes les 10 millisecondes). Un tracé dessiné entre un poser et un lever de stylo est appelé un « ***trait*** ». Un trait *'s'* est un ensemble de points définis par le quadruplet : $s=\{x(t),y(t),p(t),T(t)\}$ ; où $p$ est la pression du point et $T$ est le temps de ce point en référence avec le premier point, ces deux dernières informations n'étant pas nécessairement présentes. Ces informations reflètent la ***dynamique*** du signal de l'écriture, et permettent de calculer plus d'informations comme la vitesse et l'accélération par exemple, la Figure 4 montre un exemple de deux signaux en-ligne, tandis que 1a Figure 5 affiche les données telles qu'elles sont disponibles pour une partie du tracé.

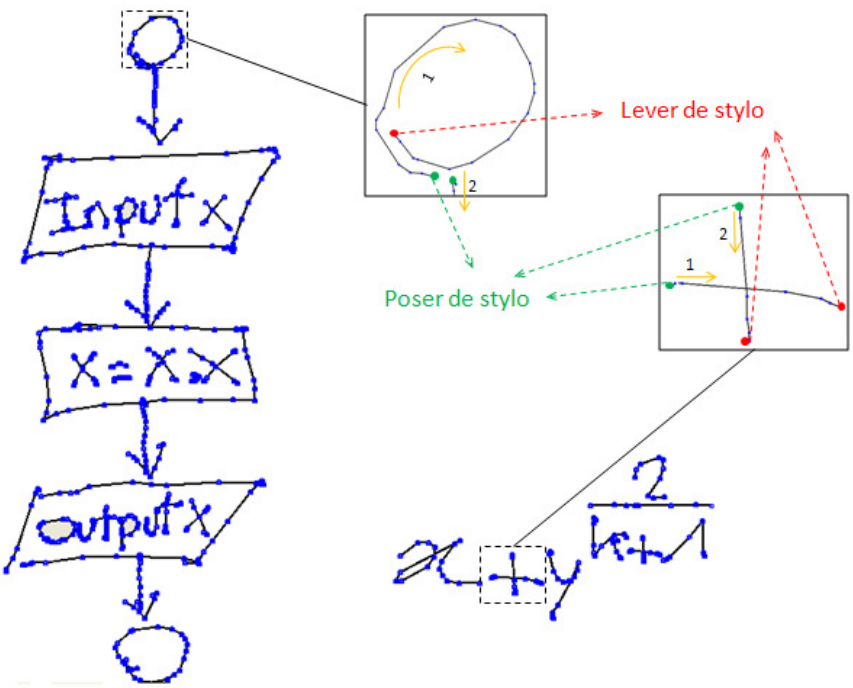



Figure 4 - Exemple du signal en-ligne d'un diagramme et d'une expression mathématique

En conséquence, il est admis que la reconnaissance de l'écriture en-ligne est plus facile que celle de l'écriture hors-ligne. Dans cette optique, des travaux ont tenté de transformer un signal hors-ligne en signal en-ligne pour profiter de ses avantages [**11**] [**12**] [**13**], cette transformation reste très difficile. L'écriture en-ligne est plutôt utilisée dans des applications temps réel, dans les domaines de la médecine, de l'éducation, des réunions, et toutes les situations qui nécessitent une prise de notes, mais aussi les interfaces homme-machine. En ce qui nous concerne, nous nous intéresserons dans ce manuscrit aux extensions de l'écriture manuscrite en-ligne aux applications de type bidimensionnelle (langage 2D).

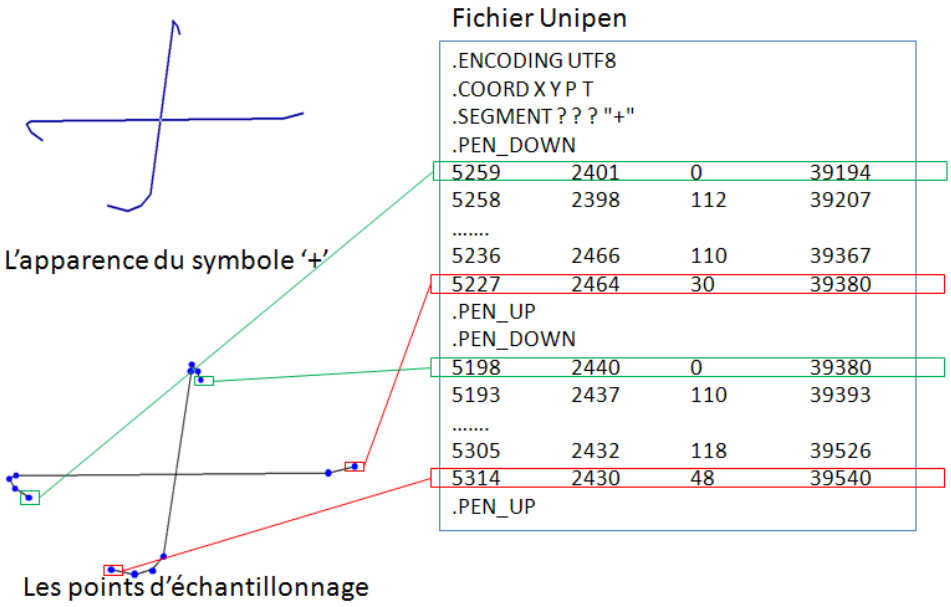



Figure 5 - Exemple d'un signal en-ligne

***A la volée Vs. a posteriori :***

Il est à remarquer que l'outil de saisie peut avoir un impact sur la structure du système et la méthodologie de reconnaissance. Les outils tels que smartphones ou tablet pc permettent une reconnaissance à la volée où l'utilisateur aura un retour visuel immédiat. Cette méthode de reconnaissance permet de simplifier quelques étapes du processus de reconnaissance (notamment la segmentation). De plus, elle introduit un grand avantage d'interactivité avec l'utilisateur : les symboles sont reconnus et interprétés au fur et à mesure puis valider (implicitement) par l'utilisateur. Ceci permet de corriger des erreurs éventuelles et de continuer l'analyse sur une base valide. Par contre, il faut qu'une méthode à la volée soit compatible avec une interaction avec l'utilisateur en temps réel.

Dans un système de reconnaissance de l'écriture hors-ligne, la reconnaissance a posteriori est imposée par défaut car l'image de l'intégralité de l'encre est déjà présente. Ceci est également vrai dans le cas d'utilisation de stylos digitaux (stylo/papier numérique), où l'encre est récupérée des stylos après la saisie. La totalité de ce qui était écrit est donc à disposition du système. La reconnaissance a posteriori s'annonce à la fois plus difficile à réaliser (car sans validation de l'utilisateur) mais en gardant l'avantage d'avoir le contexte global de l'écriture.

Dans le cadre de cette étude, nous nous intéressons à la reconnaissance a posteriori et nous considérons que tout le signal de l'écriture est présent au moment de la reconnaissance.

## 1.3. Problématiques

De nombreux outils existants permettent la saisie de tels langages 2D dans les documents numériques. Malgré la convivialité de ces outils, une certaine expertise est exigée pour les utiliser. Par exemple, pour saisir une expression mathématique dans un document, certains symboles et fonctions mathématiques sont décrits par des mots clés prédéfinis qu'il faut mémoriser, c'est le cas avec Latex et MathML. De plus, l'expression qui est naturellement bidimensionnelle doit être écrite en une seule dimension, rendant la tâche plus complexe. D'autres outils, tels que Math Type, dépendent d'un environnement visuel pour ajouter des symboles à l'aide de la souris. Ces dépendances de mots clés ou de la souris augmentent considérablement le temps requis pour saisir une expression.

D'autre part, les outils de saisie de graphiques (schémas électriques, diagrammes, …) dépendent de l'ajout via des menus des symboles graphiques disponibles pour l'utilisateur, et de l'établissement des liens entre ces symboles, et éventuellement de l'ajout du texte explicatif à l'aide du clavier. Là aussi, cela nécessite un temps important pour réaliser des graphiques complexes. Dans ce conditions pour la plupart des utilisateurs, il est beaucoup plus convivial de prendre un crayon et de dessiner sur une feuille de papier ou directement sur un écran. Tenant compte de ces considérations, il est naturel de concevoir un système capable de transformer un graphique de son état manuscrit à un format numérique. L'avantage d'un tel système est que l'on extrait en plus de la description physique du graphique, une autre description logique. Cette dernière peut être très utile pour alimenter des systèmes de calcul ou des systèmes de dessin par ordinateur (CAD). Car même si on fait la conception en utilisant un des systèmes existants de saisie, il reste dans la plupart du temps une étape humaine pour extraire des informations pertinentes [**14**].

Une question très importante se pose au moment de la conception d'un système de reconnaissance d'un langage 2D : quelle liberté faut-il donner à l'utilisateur ? Un système très contraignant facilite la tâche de reconnaissance, mais rend le système moins confortable à utiliser. Par exemple, on peut imaginer un système qui ne permet d'entrer les symboles que les uns après les autres d'une façon bien distincte. Dans ce cas, on rend trivial l'étape de segmentation qui est d'habitude très difficile à réaliser.

Au contraire, un système sans contraintes est beaucoup plus naturel à utiliser mais beaucoup plus complexe à réaliser. La reconnaissance des langages 2D est un vrai challenge qui nécessite le mariage de compétences de plusieurs domaines. La difficulté de reconnaître des langages bidimensionnels peut se résumer dans les points suivants : la segmentation, la classification et l'interprétation.

***La segmentation* :**

Un enjeu important est de pouvoir bien segmenter le graphique en ses éléments de base. Dans un signal en-ligne on peut imposer un lever de stylo entre les symboles. Cette condition est admissible seulement dans certains langages. C'est le cas pour les notations mathématiques [**15**], les équations chimiques [**7**], les organigrammes [**4**], etc. Cette hypothèse ne résout pas le problème de la segmentation mais le facilite énormément. Il suffira donc de trouver les regroupements de tracés qui appartiennent au même symbole. Toutefois, il n'est pas trivial de trouver ces regroupements, comme nous allons le voir. Quant aux autres langages, il est possible que tout ou partie du graphique soit fait avec un seul tracé. Dans ce cas il s'agit de trouver les meilleurs points de découpage entre les symboles, ce qui est loin d'être évident. La Figure 6 montre un exemple d'une équation chimique où les traits d'une même hypothèse de symbole ont la même couleur. Même si les symboles sont bien distincts dans l'équation, on voit bien qu'il n'est pas toujours trivial de décider comment grouper les traits d'un même symbole. Les deux segmentations sont correctes syntaxiquement. Par contre, certaines segmentations ne sont pas correctes sémantiquement, comme c'est le cas de la deuxième interprétation. Cependant, un modèle de langage ne peut pas toujours relever ce genre de problème.

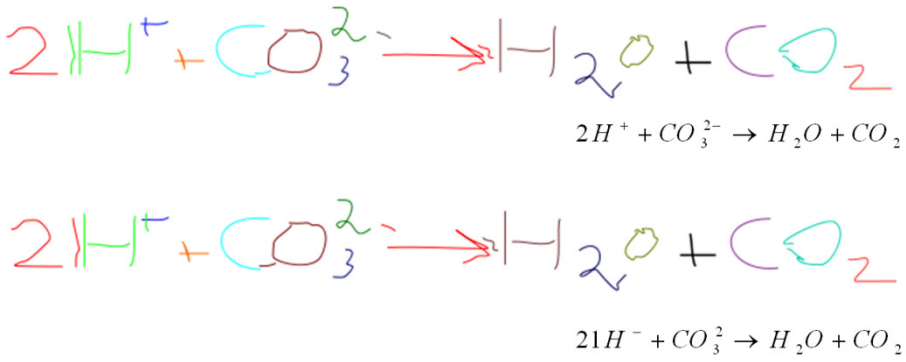



Figure 6 - Deux segmentations possibles d'une équation chimique et leurs interprétations

De même, la Figure 7 montre qu'il n'est pas évident de segmenter le tracé d'un schéma électrique manuscrit. Les carrés pointillés indiquent les zones du tracé où une coupure est possible. Il en résulte qu'il n'est pas possible de choisir d'une façon unique les points de segmentation.

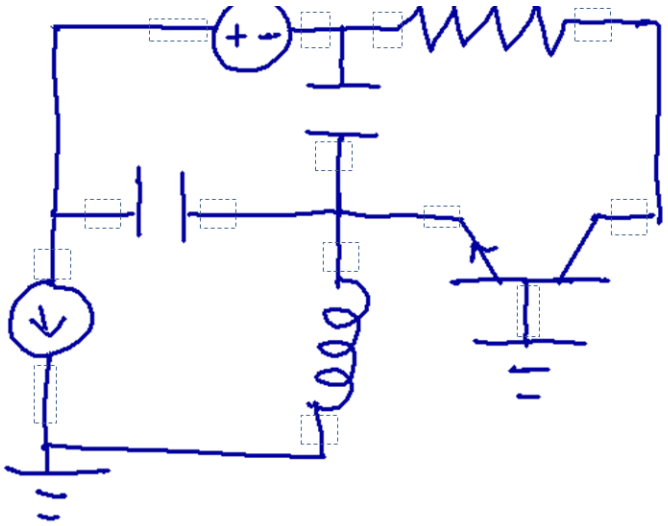


Figure 7 - Les zones admissibles de découpage d'un schéma électrique

Dee plus, avec une telle application, on ne peut pas facilement imposer une condition pour bien séparer les symboles dans un signal hors-ligne. Cela complique encore davantage la segmentation des symboles, car alors il faut à la fois isoler les composants du même symbole mais aussi détecter les symboles qui se touchent et les séparer.

Une difficulté supplémentaire s'ajoute à la segmentation quand il y a du texte explicatif dans le graphique. Car un outil performant de reconnaissance de symboles graphiques, n'est pas forcement capable de reconnaître du texte. Du coup, il faut d'abord séparer le texte du reste du graphique et ensuite envoyer chaque type de composante à son propre reconnaisseur. Une solution est de demander à l'utilisateur d'indiquer explicitement le mode de saisie grâce à un commutateur [**4**] entre les modules de reconnaissance, mais cette contrainte n'est pas très confortable et restreint sa liberté surtout quand il y a beaucoup de textes présents comme c'est le cas dans les organigrammes, les schémas électriques, etc.

***La classification de symboles***

Le deuxième défi dans un système de reconnaissance de langage 2D est le classifieur de symboles. La classification des symboles est l'étape la plus importante mais pas forcément la plus difficile grâce à la richesse de la littérature du domaine de la reconnaissance des formes.

Toutefois, la nature de ces langages 2D pose certaines spécificités qui doivent être prises en compte pour concevoir le système idéal de reconnaissance. Le classifieur doit prendre en compte la variabilité de l'écriture manuscrite. Le même symbole peut s'écrire différemment d'un scripteur à un autre. On doit également permettre l'écriture des symboles en plusieurs tracés dans n'importe quel ordre. De plus, quelques langages 2D comportent un très grand nombre de symboles (par exemple plus de 220

symboles dans la notation mathématique). Certains de ces symboles peuvent jouer des rôles différents selon leurs positions et le contexte où ils se trouvent. En outre, le même symbole peut être présent dans un graphique avec des tailles et des orientations différentes. Il est donc généralement indispensable d'appliquer une étape de prétraitement et de normalisation pour rendre la classification plus robuste.

***L'interprétation***

L'interprétation vise à trouver le sens du graphique à reconnaître. Elle consiste généralement en deux étapes : l'analyse structurelle et syntaxique. L'objectif est dans un premier temps de corriger des erreurs éventuelles dans la segmentation ou la reconnaissance. Deuxièmement, l'interprétation permet de trouver la structure du graphique, ce qui est indispensable pour transformer le graphique en format numérique. L'analyse structurelle permet d'extraire des informations spatiales (géométriques) des symboles, et les relations spatiales les reliant (topologie, distance, …). Ces informations alimentent le module d'analyse syntaxique où l'on applique les règles qui conviennent pour justifier la disposition spatiale des symboles.

Dans les structures de type schéma électrique, diagramme, organigramme, etc. il peut être intéressant d'utiliser des contraintes de connectivité entre les symboles et les connecteurs. Ainsi, treize contraintes de connectivité [**16**] basées sur une connaissance a priori ont été recensées pour un type de schémas électriques. Dans cet exemple, quand on reconnaît une résistance horizontale, il faut trouver deux connecteurs l'un à sa gauche, l'autre à sa droite. Par contre, cette approche n'est pas capable de fournir la description syntaxique du graphique, or celle-ci est nécessaire si l'on veut effectuer des opérations de haut niveau sur le graphiques. Par exemple, avoir la syntaxe du schéma permet de le résoudre et de trouver des valeurs inconnues en utilisant un solveur approprié.

Un autre exemple est celui des partitions musicales qui comportent des dépendances contextuelles locales et globales ce qui augmente la difficulté de l'interprétation. Une grammaire d'attributs peut être utilisée dans ce cas pour interpréter les partitions. Elle permet de trouver la structure de la notation et aussi les attributs de chaque partition. La règle la plus simple d'une telle grammaire décrit les symboles musicaux comme étant composés de primitives. D'autres règles de plus haut niveau sont utilisées pour la représentation des partitions. Grâce aux règles existantes de composition des partitions musicales, des contraintes contextuelles facilement applicables dans des modèles linguistiques ont été proposées [**8**].

Prusa et al. [**17**] ont présenté la possibilité de généraliser le concept de grammaire hors-contexte (CFG : Context Free Grammar) aux langages bidimensionnels. Cette approche a été particulièrement utilisée pour la reconnaissance d'expressions mathématiques [**18**] [**19**]. Une autre approche est

basée sur la théorie des graphes, puisque l'on peut naturellement représenter un graphique 2D par un graphe. Des règles de réécriture de graphe ont été employées avec succès pour la reconnaissance d'expressions [**20**]. D'autres variantes de modèles linguistiques ont été proposées pour la reconnaissance d'expressions (voir chapitre 2).

Nous pouvons constater de ces travaux que l'interprétation est le vrai défi de la reconnaissance de structures 2D. En effet, analyser les langages bidimensionnels est une tâche complexe, c'est pourquoi les travaux présentés dans [**21**] s'intéressent à des algorithmes spéciaux et à l'ajout de contraintes pour diminuer leur complexité.

## 1.4. Motivations

Nous avons choisi de traiter le problème de la reconnaissance d'expressions mathématiques manuscrites en-ligne. D'une part, la notation mathématique est un exemple de langage 2D très riche. D'autre part, l'utilisation de notations mathématiques est indispensable dans la documentation scientifique. Les expressions mathématiques représentent un outil universel de communication entre les scientifiques. Bien que ces expressions puissent apparaître compliquées pour les non-spécialistes, elles décrivent parfaitement les problèmes dans la plupart des domaines. Pour mesurer l'importance des expressions mathématiques, nous avons extrait toutes celles qui se trouvent dans les pages web de « Wikipedia » français. Un ensemble de près de 77 000 expressions (plus ou moins complexes) ont été retrouvées dans 7 000 pages web. Nous allons montrer plus tard dans ce manuscrit que le même système conçu pour la reconnaissance d'expressions mathématiques a été facilement adapté à d'autres langages 2D, comme les organigrammes de programmation par exemple. Nous allons présenter dans les paragraphes qui suivent la notation mathématique et ses spécificités qui rendent la reconnaissance d'expressions un vrai challenge.

## 1.5. La notation mathématique

L'utilisation de la notation mathématique remonte à des milliers d'années dans l'histoire. Au début, elle était présente simplement dans des systèmes de numération. Mais depuis, les savants ont employé les mathématiques et les ont développées au fur et à mesure de leurs besoins scientifiques. Ces développements ne concernaient pas seulement les symboles utilisés pour décrire des problèmes mathématiques, mais aussi la science des mathématiques elle-même. Cela a conduit à ce qu'on appelle aujourd'hui la notation mathématique moderne [**22**]. De nos jours, la notation mathématique est largement utilisée dans les domaines de la physique, la chimie, l'ingénierie, la médecine, l'économie, etc. Autrement dit, elle est utilisée dans presque toutes les sciences.

### 1.5.1. Définitions

La notation mathématique est un système d'écriture qui décrit des problèmes et des concepts d'une façon abstraite quelque soit le domaine concerné. Dans ce manuscrit nous allons nous baser sur les définitions suivantes qui caractérisent les propriétés générales de la notation mathématique moderne (d'un point de vue d'un système de reconnaissance).

***Expression mathématique***

Une expression mathématique est un ensemble de symboles mathématiques arrangés dans un espace bidimensionnel, qui a un sens sémantique. Pour la simplification dans ce manuscrit les deux termes 'expression' et 'expression mathématique' sont équivalentes. Une ***sous-expression*** est une partie de l'expression ayant une interprétation sémantiquement complète. Une ***équation*** mathématique dénote simplement une égalité entre deux ou plusieurs sous-expressions, on l'appelle généralement une expression aussi. Par exemple, l'expression $a^{b+c} = (x+y)^2$ est une équation composée de deux sous-expressions $a^{b+c}$ et $(x+y)^2$.

***Symbole mathématique***

Un symbole mathématique est l'élément de base dans une expression mathématique. Il porte un sens différent selon sa classe et sa position dans l'expression. Par exemple, les symboles présents dans l'expressions ci-dessus sont {*a, b, c, x, y, 2, +, =, (, )*}.

***Relation spatiale***

Une relation spatiale permet de décrire les positions relatives entre deux symboles. Par exemple, étant donnée l'expression *ab* les symboles *a et b* sont connectés par la relation gauche/droite. Alors qu'ils sont reliés par la relation (haut-droit/bas-gauche) dans l'expression $a^b$. Dans la plupart des systèmes proposés une relation porte en plus de son sens structurel un sens sémantique (***relation logique***). Par exemple, la relation structurelle dans $a^b$ signifie la relation mathématique *'a à la puissance b'*. De plus, une expression plus complexe peut contenir des relations sous-expression/symbole, symbole/sous-expression, ou sous-expression/sous-expression.

### 1.5.2. Spécificité des expressions mathématiques

La difficulté de la reconnaissance d'expressions mathématiques est héritée de celle des structures bidimensionnelles. Mais il y a quelques difficultés et ambiguïtés qui sont spécifiques aux expressions mathématiques. Il y a deux sources principales d'ambiguïté : le grand nombre et la variété des symboles, et la disposition bidimensionnelle de ces symboles.

#### 1.5.2.1. Le nombre des symboles

Le nombre de symboles mathématiques se trouve être relativement élevé comparé à du simple texte où moins de centaine de symboles peut suffire avec un alphabet latin (2 x 26 caractères, plus quelques caractères spéciaux et la ponctuation). De même, les applications de reconnaissance de schémas électriques ou de reconnaissance d'organigrammes nécessitent quelques dizaines de symboles. Sans vouloir prétendre à l'exhaustivité, les champs disciplinaires ayant tous leurs symboles spécifiques, nous avons recensé près de 220 symboles permettant de couvrir un large corpus d'expressions.

Les symboles sont divisés en sous-groupes selon leurs tâches et leurs natures : chiffres, alphabet (grec, latin), opérateurs, flèches, symboles élastiques, et fonctions. Nous allons détailler ces groupes et leurs caractéristiques dans la description de notre base de symboles isolés dans le chapitre 3.

Parmi ces symboles, il existe de nombreux ***recouvrements inter-classes***. La Figure 8 montre la difficulté de faire la différence entre les cas minuscule et majuscule de quelques lettres de l'alphabet latin (O et o par exemple). Contrairement au texte, les lettres majuscules se trouvent n'importe où dans une expression mathématique. Sans information additionnelle du domaine ou du problème présenté dans l'expression, il est presque impossible de distinguer les deux cas. Sur la Figure 8 il est facile de distinguer l'état des lettres *(a, B, Q)* ; mais l'état des autres lettres *(c, o, p)* n'est pas clair du tout. De plus, la nature de l'écriture manuscrite ajoute une grande variabilité à la taille des symboles. Pour éviter ce problème, on peut par exemple imposer une façon spécifique pour indiquer au système que la lettre est en majuscule [**23**].

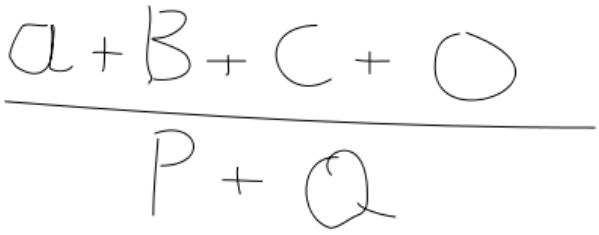


Figure 8 - Ambiguïté entre l'état minuscule et majuscule de quelques symboles mathématiques

Il existe aussi une grande ***ressemblance*** entre quelques symboles. Quelques exemples de symboles qui se ressemblent sont montrés dans la Figure 9. Là aussi, il semble très difficile de faire la différence sans avoir des informations supplémentaires.

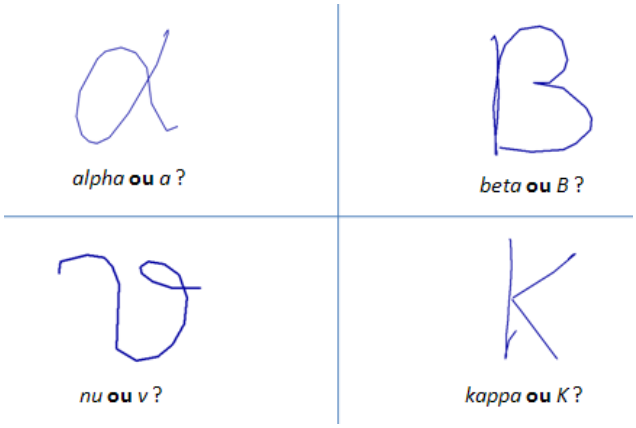



Figure 9 - Ambiguïté de ressemblance entre symboles

La plupart des symboles ont une signification bien précise. Ainsi un '3' veut toujours dire trois. Cependant il existe quelques symboles qui portent quelques ***ambiguïtés de rôle***. La Figure 10-(a) montre un exemple d'ambiguïté entre la lettre *'c'* et une parenthèse gauche *'('* [**24**]. Il s'agit dans ce cas de deux symboles au sens totalement différents qui peuvent s'écrire d'une façon identique comme par exemple le *'x'* ou le signe de multiplication, le signe du moins *'-'* ou la barre de fraction. Ils peuvent définir le même symbole mais dans des contextes différents, comme un point peut représenter un point pour les décimales, l'opérateur de multiplication, ou même n'être que du bruit à ignorer.

Néanmoins, ce genre d'ambiguïté est locale et dans la plupart des cas résolvable grâce au contexte global de l'expression gérée par la grammaire. Dans le premier cas de la Figure 10-(b), la grammaire peut ne pas supporter d'avoir deux opérateurs successifs (multiplication, addition). Il est alors convenable d'attribuer la lettre *'x'* pour le symbole en bleu. Dans le deuxième cas, si la séquence chiffre/lettre/chiffre n'est pas admissible, alors le symbole est reconnu comme le signe de la multiplication.

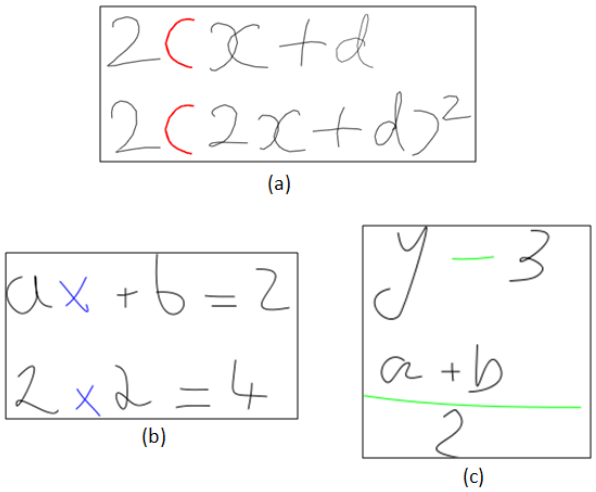



Figure 10 - Ambiguïté de rôle du symbole

Dans l'exemple (c), ce sont les contraintes structurelles qui impliquent le résultat correct de la reconnaissance. Pour attribuer une barre de fraction à une barre horizontale il faut avoir deux sous-expressions alignées verticalement au dessus et en dessous de la barre. Dans le premier cas, c'est l'alignement horizontal qui implique le choix du signe moins. Quant au premier exemple (a), l'obligation d'avoir l'ouverture et la fermeture de parenthèses favorise la lettre *'c'* dans le premier cas, et la parenthèse gauche *'('* dans le deuxième cas.

Une autre complexité vient du fait qu'il y a quelques ***symboles similaires à des sous-parties d'autres symboles***. La grammaire avec un bon classifieur et une bonne méthode de segmentation interviennent dans ce cas pour résoudre l'ambiguïté. La Figure 11 montre quelques exemples illustrant ces cas. Dans le premier exemple, le trait rouge du *'x'* peut être une parenthèse fermante. Cependant, l'ambiguïté peut être résolue car la parenthèse ouvrante correspondante n'existe pas. Dans le deuxième exemple, le trait vert ressemble à un *'c'*, mais en disposant d'un bon classifieur, on doit pouvoir avoir un bon score de reconnaissance de la fonction cosinus. Dans le dernier cas, le même trait (en bleu), est présent plusieurs fois dans l'expression. A la fois il fait partie d'autres symboles (±, =) et il est présent en tant que signe *moins*. Dans ce cas, l'ambiguïté est levée grâce aux contraintes structurelles qui interdisent l'alignement vertical de deux opérateurs et un classifieur capable de bien reconnaître les symboles (±, =).

De plus, certains symboles (ex : ∑, —, (, [,…) dits élastiques, ont des ***variations de taille*** très importantes d'une expression à une autre selon les sous-expressions qu'ils entourent.

Enfin, de ***petits symboles*** tels que les points, les virgules, les apostrophes etc. peuvent se confondre avec du bruit augmentant la complexité. Donc, il est très important de choisir une méthode adaptée de réduction de bruit lors de la phase de prétraitement.

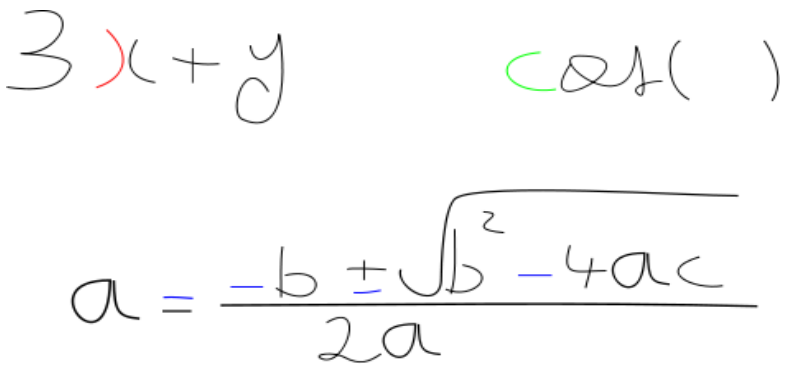


Figure 11 - Exemples de symboles inclus dans d'autres symboles

### 1.5.2.2. La disposition bidimensionnelle des symboles

Comme nous l'avons vu, les relations spatiales dans une expression mathématique fournissent une grande partie des informations utiles. Bien que les notations mathématiques soient standardisées, une grande flexibilité est permise pour le positionnement relatif des symboles.

Il en résulte que l'on ne peut pas d'une façon absolue définir une relation entre deux symboles. Comme on le voit sur la Figure 12, il n'est pas évident de choisir la relation lorsque le positionnement est aux frontières entre les relations droit, haut-droit, et haut.

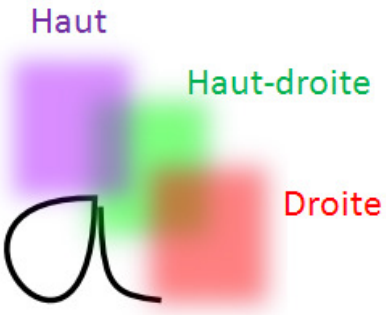



Figure 12 - Recouvrement des positions valides des relations spatiales

Pourtant c'est absolument nécessaire quand il s'agit d'opérations implicites. Ainsi les exposants, les indices, les multiplications implicites, les enchaînements de chiffres sont indiqués uniquement par l'arrangement spatial des opérandes (les symboles). Cette nature floue des relations spatiales complique considérablement la phase d'analyse structurelle. Plusieurs méthodes ont été mises en œuvre pour surmonter l'***ambiguïté des relations spatiales***, comme nous l'allons voir dans l'état de l'art (chapitre 2).

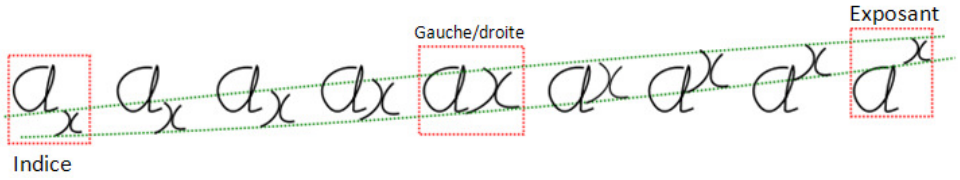



Figure 13 - Ambiguïté des relations spatiales entre deux symboles

La Figure 13 montre la nature floue des relations possibles entre deux symboles. Il est très facile de distinguer les cas extrêmes : exposant, gauche/droit, et indice. Mais dans les cas intermédiaires, la décision est moins évidente, même si l'on n'hésite à chaque fois qu'entre deux relations possibles (indice et gauche/droite ou gauche/droite et exposant). La situation se complique de plus en plus en fonction du nombre de symboles. Pour trois symboles, comme le montre la Figure 14, il y a douze situations possibles en plus de la situation initiale *abc*. Ce nombre monte très rapidement à 36 pour 4 symboles et ainsi de suite.

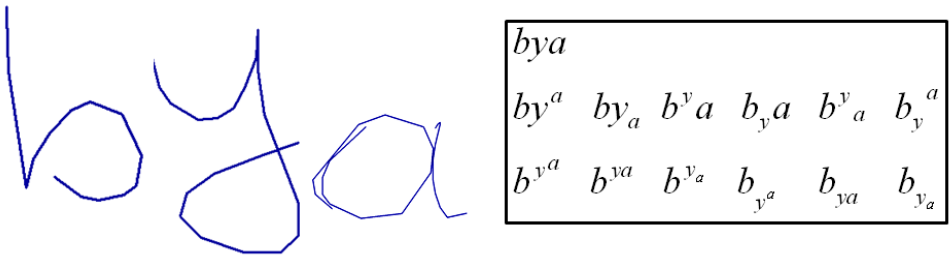



Figure 14 - Les relations spatiales possibles entre 3 symboles

Néanmoins, même si la bonne relation entre deux symboles a été trouvée, il peut être difficile d'en déduire la relation logique qui existe entre eux à cause de l'***ambiguïté du placement relatif***. Autrement dit, une relation spatiale ne peut pas être déterminée individuellement d'une façon locale. Par exemple dans la Figure 15, dans *'$b_c$'* on peut déduire localement que le *'c'* est en indice de *'b'* dans l'expression *'$b_c d$'*. Mais il se peut aussi que le *'c'* corresponde à la multiplication d'une variable à la puissance *'b'* comme dans l'expression *'$a^b c$'* [**25**]. Bien évidemment, l'information contextuelle doit permettre de résoudre ce type de problèmes.

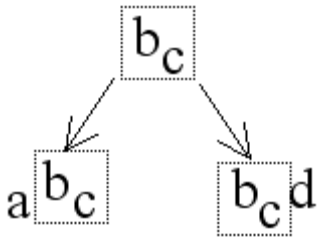


Figure 15 - Exemple d'ambiguïté du placement relatif des symboles

Toutes les ambiguïtés montrées ci-dessus sont encore plus fortes dans le cas du manuscrit car les scripteurs prennent une grande liberté de placement et d'alignement des symboles constituant l'expression.

Cependant, le contexte aide généralement à lever plusieurs ambiguïtés. Par exemple, quand on reconnaît le symbole 'somme' ∑, il n'y a que trois cas possibles. Soit c'est une somme sans limite, donc on n'attend qu'une sous-expression à droite, ou alors on s'attend à la sous-expression à droite en plus des limites au dessus et en dessous, ou simplement un indice en dessous et une sous-expression à droite.

Nous avons vu dans cette partie des problèmes spécifiques liés à la reconnaissance d'expressions mathématiques. Tous ces problèmes rendent le processus très difficile. On trouve dans la littérature beaucoup de tentatives pour concevoir un système de reconnaissance capable de surmonter autant que possible ces problèmes en limitant les contraintes, voir chapitre 2.

Nous allons dans la suite explorer quelques systèmes de la reconnaissance d'expressions mathématiques manuscrites en particulier et de structures graphiques 2D en générale.

## 1.6. Quelques systèmes existants

Nous pouvons résumer les applications potentielles de la reconnaissance de structures 2D par :

- Aide à la saisie pour l'édition de documents numériques.
- Assistance à la présentation dans le domaine de l'enseignement et des conférences en utilisant des tableaux interactifs avec une grande puissance de stockage, analyse, et réaction.
- Aide à la conception assistée par ordinateur pour des applications métiers.

Nous présentons dans la suite quelques applications qui prennent avantage des systèmes de reconnaissance de graphiques 2D.

***DMOS : Une méthode générique pour la reconnaisse de documents 2D [6]***

Coüasnon propose une méthode pour générer automatiquement des systèmes de reconnaissance de documents structurés (langage 2D). L'adaptation au nouveau langage est faite en définissant la description du document par une nouvelle grammaire : EPF (Enhanced Position Formalism). La grammaire EPF a l'avantage de décrire la structure, la syntaxe, et la sémantique d'un langage 2D. La méthode essaye de faire un compromis entre la généralité et la qualité de reconnaissance en séparant la connaissance du système et en utilisant la grammaire EPF.

On peut considérer cette approche comme une extension d'une grammaire DCG (Definite Clause Grammar) à une grammaire bidimensionnelle, où deux types de terminaux sont définis : ligne-segments et matrices de pixels. Ils permettent de représenter une partie importante des éléments qui se trouvent dans un document structuré. Le formalisme EPF a été développé pour décrire cette grammaire bidimensionnelle. Six opérateurs sont appliqués à une grammaire 1D pour qu'elle soit capable de supporter des langages bidimensionnels. Coüasnon montre la généricité de leur méthode par un générateur de systèmes de reconnaissance. Ils présentent plusieurs applications possibles générées par un simple changement de la grammaire EPF du générateur : reconnaissance de partitions musicales, d'expressions mathématiques, de formulaires et de tableaux. Bien évidement, il faut un nouvel apprentissage du classifieur pour chaque système généré.

Concernant les expressions mathématiques, il s'agit d'un système de reconnaissance d'expressions typographiées (image hors ligne). Ce reconnaisseur considère les structures suivantes : exposants, indices, fonctions, sommes, produits, intégrales, et les racines carrées. Cependant, il n'est pas reporté d'évaluation des performances de ce système.

***Scritp&Go « schémas électriques »***

Cette application est l'implémentation industrielle de la méthode « DALI » proposée dans le cadre des travaux de thèse de Sébastien Macé [**26**]. L'objectif est de fournir un outil qui facilite le dessin des schémas électriques sur le terrain dans des situations de mobilité. Elle permet la composition manuscrite de schémas électrique à la volée à l'aide d'un stylet. Pour assurer l'efficacité d'une composition rapide l'application s'appuie sur une composition mono-trait, à l'exception de quelques symboles qui peuvent être saisis en plusieurs traits. Un trait peut aussi compléter un symbole déjà interprété pour en former un nouveau plus complexe.

En ce qui concerne la reconnaissance des symboles, des heuristiques ont été utilisées pour les symboles simples (tels que les connexions). Le système RESIFCAR basé sur un système d'inférence floue [**27**] [**28**] est utilisé pour la classification des symboles plus complexes. L'interprétation des schémas est

basée sur une Grammaires de Multi-Ensembles à Contraintes Pilotées par le Contexte, ou GMC-PC proposée dans [**26**] (cf. section 2.4). Cette grammaire est un nouveau concept de langage visuel (2D) avec une propriété de généricité permettant de l'utiliser pour la composition d'autres langages 2D.

***Presto : un éditeur de partitions musicales manuscrites [29]***

Le système Presto est un compromis entre la précision de la reconnaissance et la vitesse de la composition. Pour s'approcher plus d'une entrée stylo-papier, le système favorise la vitesse de réaction par rapport aux autres propriétés. Les symboles musicaux sont remplacés par des gestes simples, ce qui augmente la vitesse de saisie et facilite le processus de classification des symboles. Pour cela, seuls les symboles les plus communs sont éditables par des gestes manuscrits mono-trait simples. Seulement 7 symboles parmi 22 considérés par le système sont représentés par des gestes. Cette simplicité facilite la tâche du classifieur de symboles et augmente le confort pour l'utilisateur.

Néanmoins, l'objectif principal de ce prototype est de mesurer l'efficacité de composition des partitions musicales en utilisant le système par rapport à une saisie conventionnelle. Les expérimentations réalisées par des musiciens professionnels ont montré un bon potentiel en termes de temps nécessaire pour la composition.

***Tahuti : un système de reconnaissance de diagrammes de classes d'UML***

Tahuti est un système de reconnaissance de graphiques géométriques dédié aux diagrammes de classes [**30**]. L'avantage du système proposé est de pouvoir dessiner les objets graphiques et les connexions, en plus de gestes d'édition en toute liberté sur un tablet PC ou un tableau blanc. La fonctionnalité de ce système s'approche plus de la réalité par rapport aux travaux précédents car il s'appuie sur la géométrie des symboles graphiques et non pas sur des gestes ou formes prédéfinis.

Tahuti se base sur une structure multi-couches permettant la reconnaissance des symboles multi-traits. Il consiste en quatre étapes principales : prétraitement, sélection, reconnaissance, et identification.

Tout d'abord, les traits sont traités à la volée. Dès qu'un trait est saisi, il est transformé en l'une des primitives suivantes : ligne, ellipse, polyligne, ou une forme complexe (un ensemble de lignes et de courbes de Bézier). En plus des informations originales, chaque trait porte les interprétations possibles et leurs probabilités. Ensuite, le trait est possiblement fusionné avec des traits qui ne sont pas encore reconnus pour former une collection de traits. La collection est envoyée au classifieur afin de décider si elle présente un symbole ou une commande d'édition. A chacune des collections sont associées une ou plusieurs interprétations. Le classifieur de symboles utilise un algorithme de reconnaissance basé sur les informations géométriques des collections.

Idéalement, toutes les combinaisons possibles sont testées. Néanmoins, pratiquement ce n'est pas possible et donc une contrainte de nombre maximal de traits par collection est imposée. De plus, on considère qu'un symbole ne se fait qu'avec des traits successifs qui sont assez proches. Finalement, une interprétation complète est choisie en fonction des probabilités obtenues lors de la phase de reconnaissance de telle sorte à maximiser la probabilité globale.

Les informations contextuelles sont extraites lors de la reconnaissance des flèches (les connexions). Le type de flèche est déterminé en fonction des types des objets qu'elle relie. Puis, une association logique est créée entre les deux objets et ainsi de suite pour toutes les flèches. Même si les informations contextuelles portées par les connexions sont locales, elles sont suffisantes pour bien interpréter un diagramme de classes car les informations qui comptent sont celles qui associent les objets entre eux.

Le système ne prend pas en compte la reconnaissance du texte explicatif dans le diagramme : il se contente de l'identifier en tant que texte et de le relier avec son objet. L'identification du texte se base sur des informations géométriques qui le distinguent des symboles graphiques.

L'interface permet de manipuler le diagramme saisi facilement. On peut déplacer ou supprimer des objets ou des associations. Dans tous les cas l'utilisateur obtient une rétroaction directe tout en visualisant l'encre saisie et son interprétation. Malgré la facilité de saisie des diagrammes de classe dans l'interface, l'inconvénient de ne pas reconnaître le texte augmente le temps nécessaire pour entrer un diagramme, car les informations textuelles doivent être saisies manuellement à l'aide d'une autre interface visuelle. De plus, la structure proposée est un peu spécifique à la reconnaissance des diagrammes de classes où peu d'informations contextuelles sont nécessaires.

***Le système E-Chalk***

Taipa et al. ont proposé un système de reconnaissance d'expressions mathématiques manuscrites embarqué dans un tableau interactif [**31**] (Electronic Chalkboard E-Chalk), voir Figure 16. Le but de ce tableau est de transmettre et de stocker des cours ou des exposés en direct.

Figure 16 - Le système E-Chalk

La reconnaissance d'expressions mathématiques se fait en deux étapes : la reconnaissance des symboles isolés, et ensuite l'analyse structurelle.

Les symboles fournis au système sont divisés en trois groupes selon le nombre de traits : les symboles à un, deux, ou plus de traits. La qualité des traits est améliorée en utilisant des méthodes classiques telles que : ré-ordonnancement des traits, ré-échantillonnage, interpolation, et normalisation. On extrait ensuite les caractéristiques du trait, ce sont des caractéristiques locales (les coordonnées, la courbure) et d'autres globales concernant la topologie du trait. Pour la classification des symboles isolés, ils ont choisi un SVM (Support Vector Machine) basé sur la technique DAG (Directed Acyclic Graph, SVM-DAG) avec des noyaux à base radiale.

Quant à la description structurelle, deux types de composants sont considérés : les symboles et les lignes de base. Les lignes de base sont des fils des symboles dans une structure de type BST (Baseline Structure Tree : arbre structurel de lignes de base). Ils décrivent une région de l'expression et la relation entre un symbole et sa ligne de base. L'idée principale du système proposé est d'utiliser les informations sémantiques des symboles pour les regrouper dans le BST qui décrit au mieux la structure de l'expression. Pour cela, on représente les symboles par des nœuds d'un graphe où un arc entre deux symboles représente la relation sémantique entre eux. Un arbre couvrant de poids minimal (MST : Minimum Spanning Tree) est calculé pour ce graphe en appliquant l'algorithme de Prim pour retrouver la description structurelle de l'expression.

L'inconvénient principal de ce système est la séparation entre l'étape de reconnaissance de symboles et l'analyse structurelle. En conséquence, il faut entrer les symboles bien séparés temporellement pour qu'ils soient bien reconnus. Le BST et d'autres structures vont être présentés dans l'état de l'art

de ce manuscrit. C'est également le cas pour les principaux classifieurs, notamment ceux de type SVM.

***MathPad² : un système de création et d'exploration d'illustrations mathématiques***

Viola et al. présentent un prototype d'une application pour la résolution de problèmes mathématiques [**32**]. L'interface incorpore un nouveau paradigme gestuel pour permettre l'entrée d'expressions mathématiques et de diagrammes libres manuscrits, voir Figure 17. MathPad² permet une compréhension plus profonde des problèmes mathématiques en faisant le lien avec des illustrations interactives. Le principe central de conception de ce système est d'être largement applicable à un vaste horizon de problèmes sans connaissance embarquée dans le système. Le comportement du système est donc spécifié par la définition mathématique manuscrite fournie par l'utilisateur.

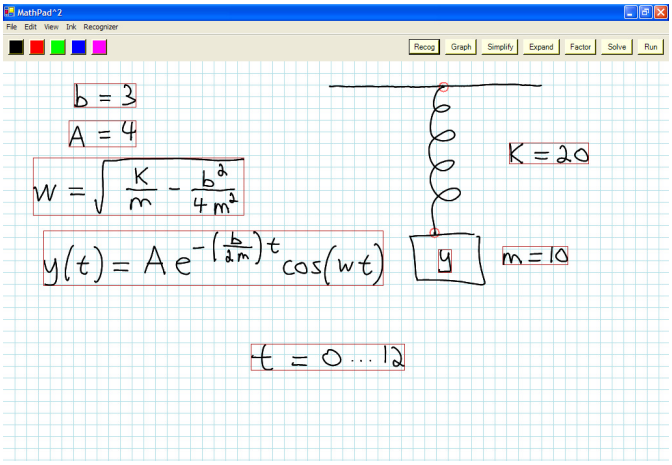



Figure 17 - Un exercice mathématique d'oscillateur harmonique

L'interface permet de faire des illustrations mathématiques en associant les expressions mathématiques et les diagrammes. Cette opération, appelée association, est faite soit implicitement en utilisant des étiquettes sur les diagrammes, soit en faisant des gestes simples dans l'interface.

Chacune des expressions mathématiques est déterminée explicitement par l'utilisateur. Une rétroaction est fournie directement pour chaque expression reconnue pour que l'utilisateur puisse corriger des erreurs éventuelles. Ensuite, l'expression reconnue est transformée en format 1D d'expressions spécifiques à Matlab (le solveur d'expressions). Dès que toutes les expressions sont reconnues, on applique la phase de rectification qui implique de définir les correspondances entre les expressions et les diagrammes pour afficher des illustrations significatives. Par exemple, si un angle d'un diagramme est associé à une valeur numérique, il est alors ajusté pour qu'il corresponde bien à la bonne valeur.

Une dernière brique du système est le moteur d'animation. D'abord, on initialise l'animation en dessinant un diagramme et son expression correspondante. On vérifie ensuite que le diagramme contient des éléments mobiles. Puis, on extrait le temps initial et final afin de calculer les positions temporelles de l'objet à animer. Les animations supportées par l'interface sont simplement des translations *(x,y)*. Cette dernière étape est très intéressante pour l'utilisateur car elle lui permet de vérifier visuellement la justesse de sa description mathématique d'un problème donné.

Malgré la restriction de ce prototype à certaines applications, ce type d'outils est très intéressant pour assister la formulation mathématique des problèmes. Néanmoins, le système proposé ici est dépendant du scripteur. Il nécessite donc une phase d'apprentissage pour chaque nouvel utilisateur. Utiliser un classifieur omni-scripteur peut rendre le système plus portable et plus facile à utiliser en classe par exemple.

***PenCalc : une nouvelle application de la technologie de reconnaissance d'expressions mathématiques manuscrites***

La plupart des calculatrices embarquées dans les appareils à base de stylo digital ne prennent pas grand avantage des technologies de reconnaissance. L'entrée d'expressions est simplement faite à l'aide d'un clavier virtuel. Chan et al. [**33**] ont proposé une calculatrice intelligente à base d'écriture manuscrite, permettant l'entrée d'expressions en les écrivant directement sur l'écran par un stylo. De plus, l'utilisateur peut définir des variables pour stocker des résultats intermédiaires. Pour la rendre efficace, il faut qu'elle soit impérativement associée à un bon système de reconnaissance d'expressions.

Le reconnaisseur implémenté utilise un classifieur de symboles qui est à la fois simple mais aussi robuste. Basé sur une approche structurelle flexible de mise en correspondance, ce classifieur est raisonnablement rapide, précis, et assez peu sensible aux variations de l'écriture. De plus, il est facilement extensible à un nouveau vocabulaire. Quant à l'analyse syntaxique, on applique une décomposition hiérarchique pour obtenir la structure syntaxique de l'expression d'une façon efficace. Finalement, un mécanisme de détection et correction d'erreurs est appelé afin de rendre le processus plus robuste.

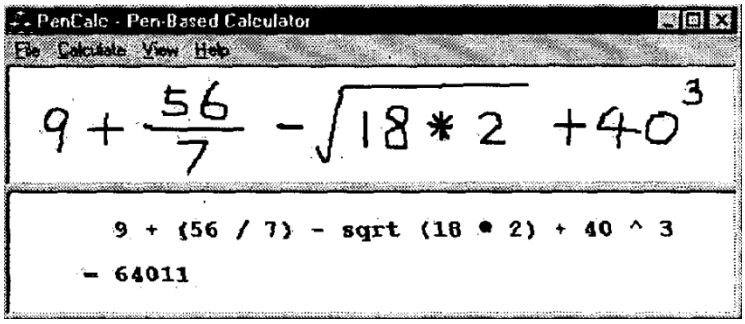



Figure 18 - Une capture d'écran de l'interface du système PenCalc

L'interface est très facile à utiliser, on écrit simplement l'expression à calculer comme on le fait sur un papier et le résultat s'affiche tout de suite. Une variable *v* est définie facilement en écrivant simplement une équation comme : *v=1000*. On pourra ensuite utiliser *v* à tout moment avec sa valeur *1000*. Toutefois, les types d'expressions acceptées sont limités à l'arithmétique basique, les nombres réels, fractions, pourcentages, exposants, et racines carrées. Un système similaire a été proposé dans [**34**] pour transférer le résultat de reconnaissance à un système de calcul algébrique.

## 1.7. Conclusion

La reconnaissance de structures bidimensionnelles est un challenge qui a motivé les chercheurs pour proposer des solutions tout au long des quatre dernières décennies. Nous avons présenté dans ce chapitre le problème de reconnaissance de langages bidimensionnels. Quelque soit le type de langage à reconnaître (schémas, diagrammes, partitions musicales, …), le système de reconnaissance peut être décrit par le schéma montré à la Figure 19.

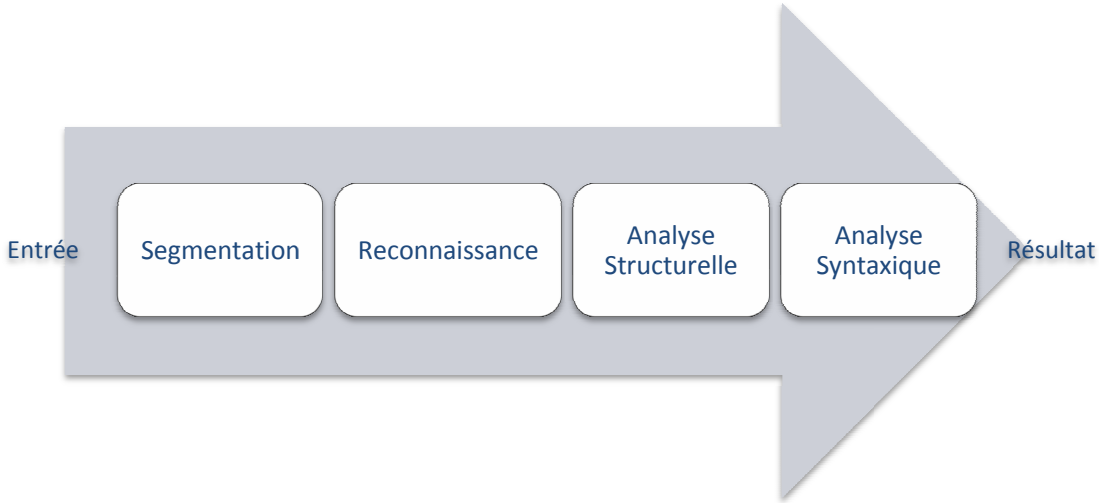



Figure 19 - Schéma global d'un système de reconnaissance de structures 2D

L'étape de segmentation vise à regrouper les traits qui appartiennent au même symbole et ensuite les reconnaître par le classifieur. Ensuite, et avant d'appliquer l'analyse syntaxique, il faut trouver la description structurelle du graphique. Cette description définit les relations entre les symboles. Finalement, l'analyse syntaxique vise à résoudre les ambiguïtés et à trouver la structure syntaxique.

La reconnaissance d'expressions mathématiques est particulièrement difficile pour plusieurs raisons. Nous avons présenté les sources d'ambiguïtés présentes dans les expressions qui rendent cette tâche très compliquée.

De plus nous avons montré quelques applications qui profitent des approches de reconnaissance des structures 2D. Dans le chapitre suivant, nous allons parcourir en détails les approches et les méthodes proposées pour la reconnaissance d'expressions mathématiques.

# CHAPITRE 2 : ETAT DE L'ART

Une expression mathématique est un arrangement complexe de symboles mathématiques en deux dimensions. Certains éléments tels que les fractions, les indices et les exposants introduisent des changements de directions et/ou de tailles dans l'agencement des symboles, ce qui rend la reconnaissance d'expressions mathématiques plus difficile et complexe par rapport à la reconnaissance du texte.

Les premiers systèmes de reconnaissance d'expressions mathématiques ont été dédiés à la reconnaissance des expressions typographiées [**35**][**36**], ou manuscrites [**37**] hors-lignes (image). Depuis, les outils de saisie des expressions mathématiques ont évolué progressivement en même temps que l'avancement technologique. On dispose de nos jours d'outils de saisie très performants en termes de précision et de maniabilité, en particulier les stylos numériques. Ceux-ci donnent accès à l'expression manuscrite sous une forme dite en-ligne (séquence des points) alors que plus traditionnellement les données disponibles avec les systèmes hors-lignes, qu'ils représentent des expressions dactylographiées ou manuscrites, se présentent sous la forme d'images (tableau de pixels). Si de prime abord, les deux problèmes sont différents, traitement d'images du coté hors-ligne et traitement du signal du coté en-ligne, en réalité, dans les deux cas les mêmes sous-problèmes sont à envisager. Une méthode proposée dans le cadre hors-ligne peut facilement être adaptée pour qu'elle s'applique sur des données en-ligne et vice-versa. Pour cette raison, nous n'allons pas dans l'exploration de l'état de l'art préciser à chaque fois la nature de l'expression, sauf si cela s'avère nécessaire. Par contre, tous les travaux que nous développerons seront appliqués à la reconnaissance d'expressions mathématiques manuscrites en-ligne.

Dans une expression hors-ligne, un symbole consiste en un ensemble de pixels noirs qui ne forme pas nécessairement une composante connexe. Dans une expression en-ligne, nous définissons un ***trait*** (***stroke*** en anglais) en tant qu'un ensemble de points tracés entre un poser et un lever de stylo. Nous considérons que chaque symbole consiste en un ou plusieurs traits séquentiels ou non-séquentiels. Ces conditions d'écriture en-ligne sont quelques fois totalement relâchées. Le scripteur peut donc écrire les symboles en plusieurs traits, avec la possibilité de traits retardés, c'est-à-dire effectués après qu'un autre symbole ait été tracé. Bien entendu, certains systèmes peuvent imposer des contraintes d'ordre et de nombre de traits.

La reconnaissance d'une expression mathématique se déroule en quatre étapes principales [**3**][**38**] : la segmentation, la reconnaissance de symboles, l'analyse structurelle puis l'analyse syntaxique de l'expression. L'étape de segmentation vise à regrouper les traits (ou les pixels) qui appartiennent au même symbole. L'étape de reconnaissance attribue une étiquette à chaque segment obtenu, enfin l'interprétation revient à analyser la structure et les relations spatiales entre les symboles avant d'appliquer une analyse syntaxique pour proposer la solution

cohérente reconnue qui est généralement exprimée sous la forme d'un arbre de dérivation. L'arbre est ensuite facilement traduit en format numérique comme par exemple : Latex, MathML, …etc.

On peut distinguer trois méthodologies principales qui ont été mises en œuvre pour la reconnaissance d'expressions mathématiques, celles-ci sont illustrées à la Figure 20 :

- Application séquentielle des différentes étapes, dans ce cas les informations circulent d'une façon séquentielle d'une étape à l'autre. Chaque étape s'applique indépendamment des autres.
- Application simultanée de la segmentation et de la reconnaissance (Seg/Rec), puis de l'analyse structurelle et syntaxique (Interprétation). Cette approche s'effectue en deux grandes étapes, à savoir la reconnaissance des symboles sans segmentation explicite (on segmente en fonction de score de reconnaissance). Ensuite, l'analyse syntaxique de l'expression utilise les informations structurelles sans passer par une étape structurelle.
- Application simultanée de tous les quatre modules. Il s'agit d'une coopération entre les différents modules afin de trouver directement l'expression la plus probable. Nous allons dénoter cette approche dans ce manuscrit par 'Approche globale'.

Nous allons dans ce chapitre explorer les méthodes et les approches utilisées pour la reconnaissance des langages bidimensionnels en général. Nous focalisons plus sur ceux dédiés aux expressions mathématiques dans les trois méthodologies ci-dessus. Remarquons que le même module (ex : un classifieur de symboles) s'adapte facilement aux trois méthodologies. En effet, c'est l'interaction et la coopération entre les modules qui diffèrent. Nous allons d'abord faire référence aux méthodes et approches que nous avons adoptées sans les détailler. Une description complète et détaillée du système et les différentes approches utilisées seront présentées dans le chapitre 4.

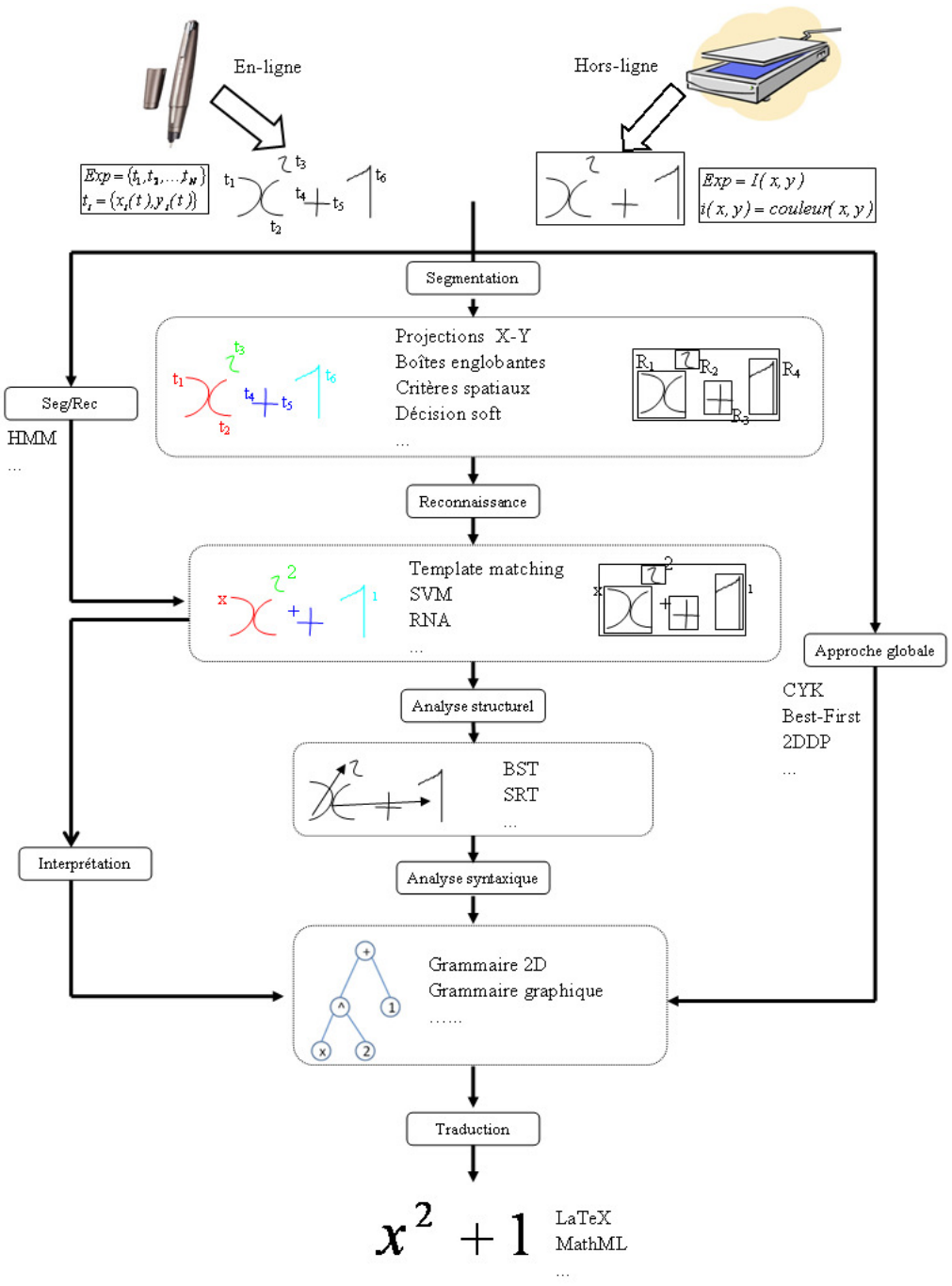



Figure 20 - Illustration du processus de reconnaissance d'expressions mathématiques

## 2.1. Segmentation des expressions mathématiques

La segmentation est une étape indispensable pour la reconnaissance de l'écriture. Quelque soit la source (en-ligne ou hors-ligne) et la nature (manuscrite ou dactylographiée) du signal de l'écriture, une bonne

segmentation des symboles est essentielle pour une bonne reconnaissance. La segmentation permet à la fois d'identifier les éléments de base qui forment le signal, et aussi d'introduire des informations spatiales indispensables dans les étapes d'analyse structurelle et de l'interprétation.

Dans un système de reconnaissance de mots non contraints [**39**], ce problème de segmentation pose déjà des difficultés évidentes. Pour les résoudre, on peut distinguer deux approches principales. L'approche globale considère le mot en tant qu'entité entière et elle esquive le problème de la segmentation. Dans ce cas, chaque mot est modélisé individuellement. Il est bien évident que la conception d'un tel système devient de plus en plus lourde lorsque la taille du lexique augmente. Dans ces conditions, une reconnaissance basée sur la segmentation (approche analytique) s'impose. Celle-ci va chercher à trouver et reconnaître les caractères du mot en question. Cette segmentation peut être explicite (INSEG), i.e. reconnaître les caractères en amont pour reconnaître le mot. Ou autrement, elle peut être implicite (OUTSEG), i.e. décider les points de segmentation en aval après avoir reconnu le mot. Bien entendu, dans ces conditions, la segmentation est rarement unique, mais elle conduit à construire un graphe ou treillis dont l'exploration combinée à la reconnaissance, permettra de trouver la meilleure hypothèse de segmentation/reconnaissance. Il est à remarquer que le texte s'écrit systématiquement de gauche à droite (ou à l'inverse dans certains scripts). Cet ordonnancement spatial facilite notamment la segmentation des mots, et même celle des phrases.

En ce qui concerne la reconnaissance d'expressions mathématiques, la segmentation est un défi plus difficile à résoudre pour plusieurs raisons. Il s'agit de résoudre le même problème mais dans un espace bidimensionnel. Les symboles mathématiques s'écrivant dans presque toutes les directions, l'hypothèse gauche-droite évoquée précédemment est ici bien trop réductrice. En outre, les symboles mathématiques peuvent avoir des tailles et positions différentes selon leurs interprétations, ce qui rend la segmentation encore plus difficile.

### 2.1.1. Cas du signal hors-ligne

Dans le cas du signal hors-ligne, la composante de base est le pixel. La phase de segmentation consiste donc à isoler les composantes (groupe de pixels) qui représentent les symboles. Un symbole se décompose en une ou plusieurs composantes connexes, par exemple « i », « j », « = », « Θ », etc. ou de formes plus compliquées comme quelques symboles en englobant d'autres (comme la racine carrée).

De nombreuses méthodes ont été proposées pour effectuer la segmentation des symboles. Faure et al. ont présenté dès le début des années 90 un système modulaire permettant de segmenter les expressions mathématiques manuscrites hors-ligne [**40**]. Ce système possède deux modules pour la

segmentation. On applique des projections successives sur l'axe des abscisses et des ordonnées pour décider comment segmenter les symboles. Mais pour assurer la segmentation de certains symboles (comme la racine carrée), on applique un masque sur l'expression afin de segmenter ces symboles et ce qu'ils contiennent avant que l'opération de projection soit effectuée sur les symboles englobés. Ensuite, des connaissances à priori sont utilisées pour corriger l'arbre de relations créé précédemment. Par exemple, des tentatives de combinaisons de différentes composantes sont mises en œuvre pour traiter les caractères comme les « i », « j » ou « = ». On peut constater sur la Figure 21 que les connaissances sont indispensables pour regrouper les segments de la somme (nœuds 1 et 2) ou ceux du x (nœuds 3 e 4).

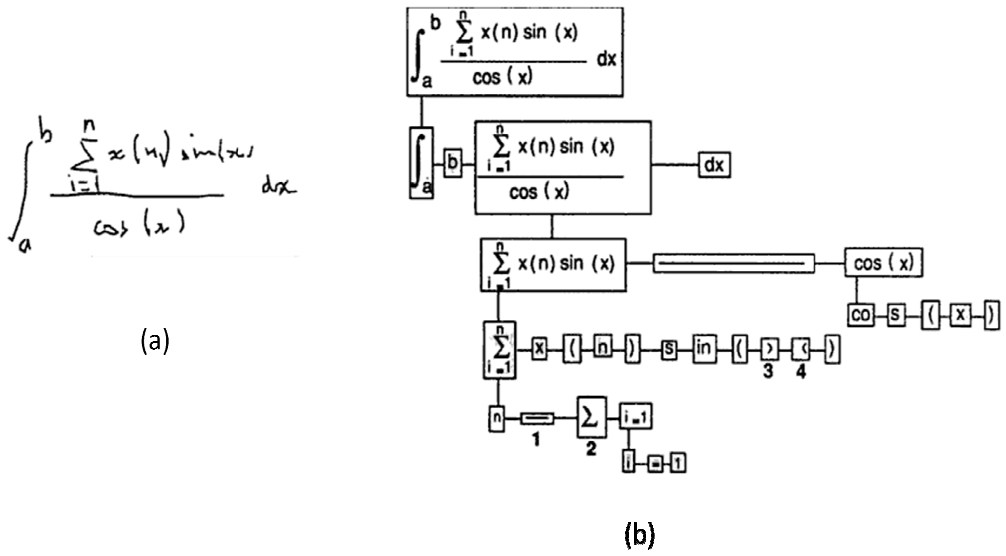



Figure 21 - (a) Image d'une expression mathématique manuscrite (b) Arbre de segmentation obtenu après avoir appliqué les projections [**40**]

Une variante de cette dernière méthode est de trouver les boites englobantes des symboles [**41**][**42**]. Les boites sont retrouvées grâce à des projections récursives sur les axes X et Y. Néanmoins, des informations à priori sont aussi nécessaires pour regrouper les composantes qui appartiennent au même symbole. L'avantage de l'approche par projections est qu'elle construit directement un arbre de relation entre les symboles. Cet arbre est ensuite utilisé pour l'analyse structurelle et syntaxique. Toutefois, ces méthodes sont handicapées pour faire face au problème des symboles qui se touchent, cela nécessitent des traitements plus élaborés pour le résoudre.

Pour éviter ces inconvénients, d'autres études ont introduit la notion de courbe comme primitive de base pour construire les symboles. Dans ce cas, une étape de prétraitement est effectuée pour trouver le squelette de l'expression. Ensuite, on considère qu'un symbole se compose d'une courbe isolée, ou de plusieurs courbes connexes. La limite de cette méthode est qu'on

ne peut pas isoler les symboles qui se composent de plusieurs composantes bien séparées, comme par exemple les « i » ou les « j » [**43**][**44**].

Comme nous l'avons vu, il est très fréquent dans un signal hors-ligne que deux symboles se touchent. La Figure 22 montre plusieurs situations possibles de symboles se touchant dans une expression hors-lignes [**45**].

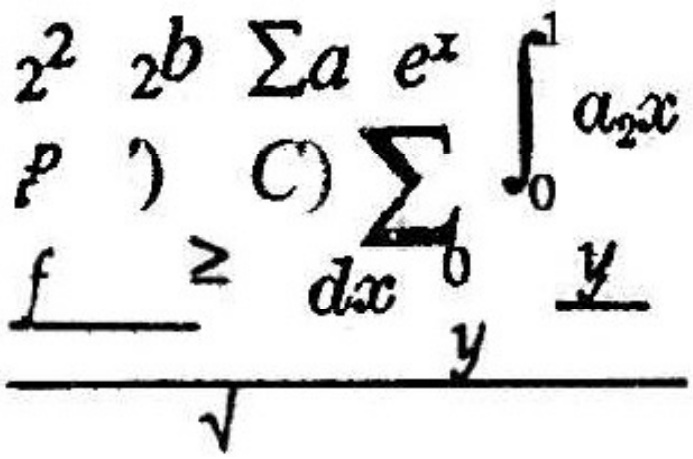


Figure 22 - Exemples de symboles qui se touchent dans un signal hors-ligne

Ces situations perturbent le processus de segmentation et doivent être impérativement détectées et corrigées. Ainsi, ce problème a été abordé dans plusieurs travaux. Deux étapes principales sont nécessaires, tout d'abord il faut détecter la présence de cette anomalie puis la corriger. Pour cela, on peut par exemple s'appuyer sur la confiance d'un classifieur [**46**][**47**].Une segmentation est considérée comme résultant d'un bloc de symboles se touchant, si son score de reconnaissance est inférieur à un seuil α. Il faut remarquer que les caractéristiques utilisées pour ce classifieur doivent être sensibles à la différence entre un symbole, et un bloc de deux ou plusieurs symboles. Dans la méthode proposée, deux types de caractéristiques sont utilisées, le rapport largeur/hauteur (ratio d'aspect) et la caractéristique périphérique [**46**]. La seconde étape consiste à trouver des mises en correspondance entre les motifs des symboles isolés et le bloc de symboles se touchant pour pouvoir ensuite les séparer. Bien évidemment cette dernière étape est coûteuse. Garain et al. [**45**] ont proposé une amélioration qui permet de détecter même des blocs à trois symboles ou plus, et d'éviter l'étape lourde de mise en correspondance avec les motifs isolés. Dans un premier temps, une analyse multifactorielle est réalisée pour détecter les blocs. Les facteurs sont extraits et calculés en se basant sur des observations liées à la nature de tels blocs dans une grande base d'expressions mathématiques. Ensuite, on peut essayer de prévoir les positions où les symboles se touchent, et puis confirmer ces positions par un classifieur. Cette dernière méthode permet d'améliorer l'efficacité de 12% par rapport à un système qui ne considère pas la détection des symboles connectés.

### 2.1.2. Cas du signal en-ligne

De l'autre coté, étant donné un signal en-ligne, nous appellerons ***trait*** un tracé réalisé entre un poser et un lever de stylo. Le trait est l'unité de base et on suppose que ce trait n'appartient qu'à un seul symbole. Cette hypothèse est peu contraignante et largement admise. Par contre, nous considèrerons dans nos travaux que chacun des symboles consiste en un ou plusieurs traits, réalisés séquentiellement ou non. Cette dernière hypothèse est rarement prise en compte, elle complique singulièrement la segmentation car alors, un graphe classique de segmentation basé sur l'ordonnancement temporel n'est plus suffisant. L'étape de segmentation vise donc à regrouper les traits qui appartiennent au même symbole. Dans la plupart des travaux rencontrés, on considère que tous les traits sont présents au moment de la segmentation. L'avantage de cette stratégie est de prendre en compte le contexte global de l'expression et d'en profiter pour améliorer la précision de la segmentation. Une autre approche consiste à faire la segmentation à la volée. Dans ce cas, les segments sont mis à jour au fur et à mesure des traits saisis. Les systèmes avec ce dernier type de segmentation fournissent un retour instantané, rendant donc ces systèmes plus interactifs.

La Figure 23 montre plusieurs regroupements possibles d'un ensemble de traits. On voit qu'il s'agit d'un problème complexe et qui devient de plus en plus difficile quand des traits retardés sont présents. Le nombre de segmentations possibles est défini par le nombre de Bell. Sur le simple exemple de la Figure 23, en considérant la présence de 7 traits, on obtient $B_7$ = 877 segmentations distinctes.

Les nombres de Bell satisfont la formule de récurrence :

$$B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k \text{ ; où } \binom{\text{n}}{\text{k}} \text{ est un coefficient binominal}$$

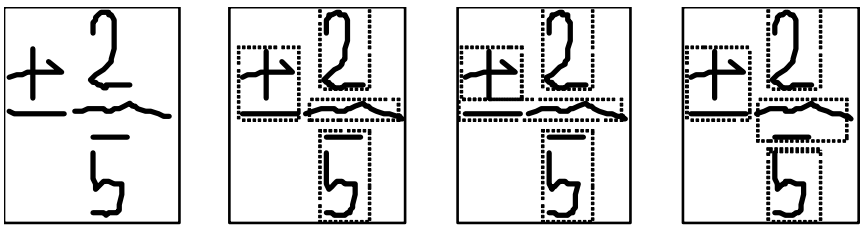


Figure 23 - Regroupements possibles d'un ensemble de 7 traits

Quelques travaux sur la reconnaissance des langages 2D proposent la suppression de l'étape de la segmentation en définissant un alphabet de symboles mono-trait. Chaque trait est alors interprété comme un symbole. C'est le cas dans les systèmes de conceptions d'interfaces graphiques SILK

[**48**] et FreeForm [**49**], le système de composition de diagramme UML Knight [**50**], et celui de composition de partitions musicales [**51**]. L'idée est de remplacer les symboles traditionnels du langage par un équivalant composé d'un seul trait. Ce principe facilite notamment le processus de reconnaissance du graphique. Mais il impose une forte contrainte aux utilisateurs, car il doit apprendre à dessiner leurs symboles par un seul trait ce qui n'est pas toujours évident, cela rend l'utilisation du système moins convivial. De plus, la difficulté est proportionnelle au nombre de symboles. Un alphabet d'une dizaine de symboles peut être facilement représentable avec des formes mono-trait. Par contre, un alphabet d'une cinquantaine de symboles mono-trait est lui-même ambiguë et peut être très difficile à retenir.

D'une façon similaire à la segmentation hors-ligne, on peut se baser sur les informations spatiales pour regrouper les traits. Une analyse ascendante attribue chaque trait à un groupe, puis chaque groupe est considéré comme un symbole [**52**]. Une méthode similaire a été proposée dans [**33**], où les traits qui ont des boîtes englobantes se chevauchant sont regroupés. En plus, les traits qui se situent dans une même fenêtre sont également regroupés. Ce dernier critère est surtout utile pour les symboles multi-traits comme « i », « j », « = ». Dans [**53**], une segmentation à la volée est effectuée en utilisant la dilatation morphologique. Quand un nouveau trait est introduit, il est regroupé avec le symbole courant si la distance du premier point est inférieure à un certain seuil. La Figure 24 montre deux cas, dans le premier cas (a), les deux traits sont regroupés car leurs dilatations sont assez proches. Au contraire, les traits dans le deuxième cas (b) sont considérés comme deux symboles différents.

Cette contrainte structurelle, dite de « proximité graphique » est également utilisée pour le regroupement de traits dans des systèmes de reconnaissance des diagrammes UML [**30**] [**54**], où les traits qui sont éloignés d'une distance inférieure à un seuil prédéfini sont regroupés en un seul symbole.


Figure 24 - Regroupement de traits en fonction de leur distance [**55**]

Lehmberg et al. [**56**] ont également adopté une méthode basée sur les informations spatiales et aussi sur des caractéristiques des traits eux-mêmes.

Un réseau de traits est construit, tel que montré en Figure 25. On associe à chaque trait $m$ une des 3 catégories possibles (primitive, standard, ou complexe) et une probabilité binaire $P_c$. Cette probabilité décide si, oui ou non, un trait peut être regroupé avec un ou plusieurs des trois traits qui suivent. Une deuxième probabilité $P_z$ est ensuite calculée pour chaque groupe de $g\ (1 \le g \le 3)$ traits. La probabilité finale de chaque groupe est donc calculée par $P(m,g) = P_c.P_z$, et un nouveau réseau de symboles est construit. Cette méthode évite le problème de critères spatiaux locaux. Toutefois, elle est très sensible à la présence de traits retardés et à l'ordre temporel des traits. Genneria et al. [**57**] proposent une méthode similaire pour la segmentation des diagrammes manuscrites (objets et connexions : les schémas électrique par exemple). Cette méthode se base sur l'idée que les traits d'un symbole portent une grande densité d'encre. A l'inverse des connexions, l'encre d'un symbole se concentre dans une petite espace.

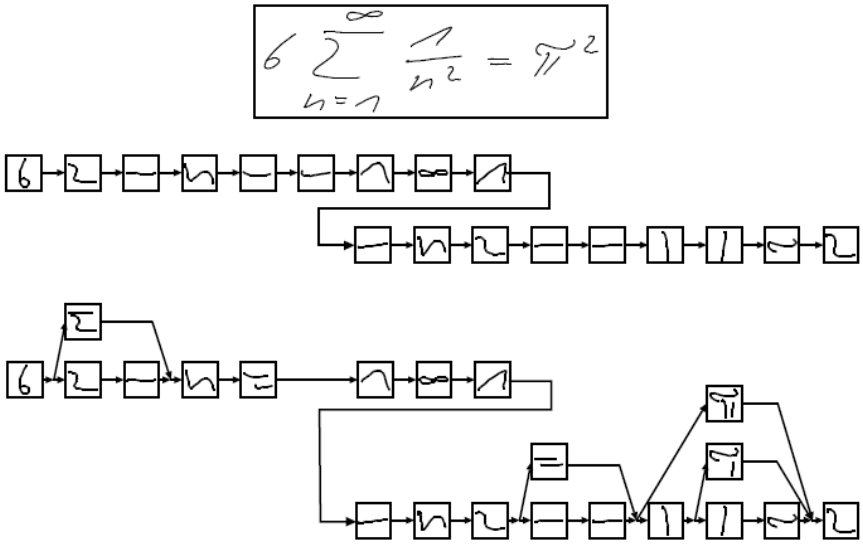


Figure 25 - Un exemple d'arbre de segmentation d'une expression en-ligne [**56**]

Cette approche a évolué vers ce qu'on appelle la segmentation et la reconnaissance simultanées. Le critère principal qui décide du regroupement de traits dans ce cas est la probabilité que ce groupe de traits représente un symbole unique. Par conséquent, les classifieurs de symboles jouent un rôle très important en même temps que les critères spatio-temporels, cf. 2.2.7.

Après avoir bien identifié les entités de base, on doit attribuer à chacune de ces entités une étiquette de symboles. Dans la suite, nous allons présenter l'étape de classification des symboles pour un système de reconnaissance d'expressions mathématiques.

## 2.2. Classification des symboles

A la fin de la phase de segmentation, on obtient une liste d'objets avec leurs attributs et caractéristiques. Ce qu'on cherche à déterminer dans la phase de classification est l'identité de ces objets. La problématique de la reconnaissance de symboles est plutôt un problème classique de reconnaissance de formes. Encore qu'elle se trouve relativement plus complexe que pour la reconnaissance de caractères dans la mesure où le nombre de classes se trouve être relativement élevé. Comme nous l'avons vu dans le chapitre 1, nous avons recensé environ 220 symboles mathématiques relativement courants permettant de couvrir un large corpus d'expressions. Parmi ces symboles, il existe de nombreux recouvrements interclasses, par exemple une barre horizontale de fraction et le signe moins [**24**]. D'autres ambiguïtés ont été présentées dans le chapitre 1. Nous allons ici présenter les différents classifieurs qui sont typiquement mis en œuvre pour la reconnaissance de symboles mathématiques ou de symboles graphiques dans des graphiques bidimensionnels.

### 2.2.1. Mise en correspondance de motifs (Template matching)

Plusieurs systèmes utilisent la méthode de mise en correspondance de motifs. Elle consiste à comparer l'objet à reconnaître avec l'ensemble des références des symboles présents dans la base de données. Les systèmes proposés dans [**41**][**58**] utilisent le « template matching » comme classifieur de symboles mathématiques typographiés. Néanmoins, ce type de classifieurs peut être également utilisé pour la classification de symboles manuscrits en-ligne [**15**]. Cette méthode reste assez lourde, malgré les nombreuses variantes pour accélérer la comparaison. Car en plus du grand nombre possible de comparaisons, il faut impérativement stocker tous les échantillons de la base de données de référence.

### 2.2.2. Approche structurelle

Cette approche consiste à extraire des primitives structurelles de la séquence de points. Les primitives sont ensuite utilisées de différentes façons pour reconnaître les symboles. Belaid et al. proposent d'effectuer la classification à l'aide d'un arbre de décisions [**37**], basé sur ces informations. Chan et al. [**59**] proposent à leur tour de modéliser les symboles en fonction de cinq types de primitives structurelles. Ces primitives sont :

- line : une ligne ;
- up : une courbe allant dans le sens contraire des aiguilles d'une montre ;
- down : une courbe allant dans le sens des aiguilles d'une montre ;
- loop : une courbe qui se rejoint elle-même en un point ;
- dot : un très petit trait.

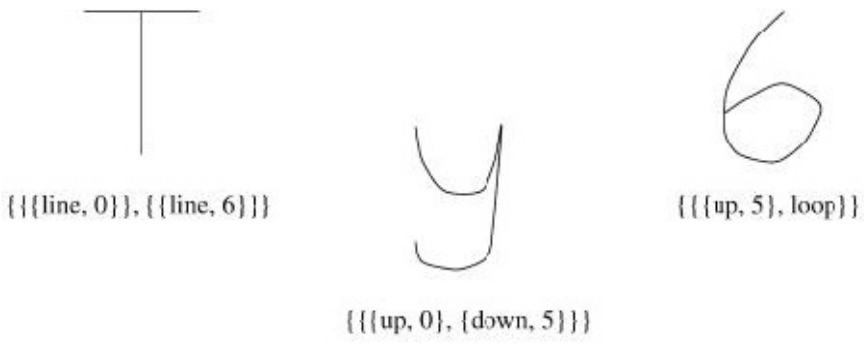



Figure 26 - Exemple de représentation de caractère avec les primitives structurelles

La Figure 26 montre un exemple des modèles structurels des symboles (T, Y, et 6). On compare ensuite ces modèles avec les modèles de la base d'apprentissage pour essayer de trouver le symbole correspondant. Afin de pallier aux variations de style et de taille des symboles, une mise en correspondance flexible de type « elastic matching » est mise en œuvre. Xu et al. proposent de représenter les primitives structurelles d'un symbole graphique par un graphe de relations spatiales [**60**]. Dans ce cas, la classification consiste à appliquer des techniques d'isomorphisme afin de trouver un « graphe modèle » le plus proche du graphe du symbole à reconnaître. La limite de telle méthode est la complexité calculatoire qui nécessite l'ajout d'heuristiques pour la rendre utilisable en pratique.

### 2.2.3. Les K-plus proches voisins (k-ppv)

L'utilisation de classifieurs basés sur la méthode des K-plus proches voisins a été proposée dans [**19**][**61**], du fait de sa simplicité et de ces performances. En effet, c'est une des plus simples méthodes de classification car elle ne nécessite pas d'étape d'apprentissage explicite. L'ensemble des connaissances du système est simplement la base d'apprentissage elle-même. L'entrée est donc attribuée à la classe la plus présente dans son voisinage de l'espace des caractéristiques. Toutefois, de même que pour la méthode de mise en correspondance, il faut stocker toute la base des données d'apprentissage, ce qui rend la classification très coûteuse en temps de calcul. En plus, la performance dépend essentiellement de la pertinence de la base d'apprentissage. Cela fait que cette méthode est moins adaptée aux systèmes multi-scripteurs.

### 2.2.4. Systèmes à vastes marges (SVM, « Support Vector Machines »)

Cette technique de classification s'est avérée très performante dans plusieurs domaines [**62**].

La première idée clé des SVM est la notion de ***marge maximale***. La marge est la distance entre la frontière de séparation et les échantillons les plus

proches. Ces derniers sont appelés ***vecteurs supports***. Dans les SVM, la frontière de séparation est choisie comme celle qui maximise la marge. Ce choix est justifié par la théorie de Vapnik-Chervonenkis (ou théorie statistique de l'apprentissage). Le problème est de trouver cette frontière séparatrice optimale, à partir d'un ensemble d'apprentissage. Ceci est fait en formulant le problème comme un problème d'optimisation quadratique, pour lequel il existe des algorithmes connus.

Afin de pouvoir traiter des cas où les données ne sont pas linéairement séparables, la deuxième idée clé des SVM est de transformer l'espace de représentation des données d'entrées en un espace de plus grande dimension (possiblement de dimension infinie), dans lequel il est probable qu'il existe un séparateur linéaire. Ceci est réalisé grâce à une fonction noyau, qui doit respecter certaines conditions, et qui a l'avantage de ne pas nécessiter la connaissance explicite de la transformation à appliquer pour le changement d'espace. Les fonctions noyaux permettent de transformer un produit scalaire dans un espace de grande dimension, ce qui est coûteux, en une simple évaluation ponctuelle d'une fonction. Cette technique est connue sous le nom de kernel trick.

L'inconvénient majeur de cette méthode est le coût important de mémorisation des vecteurs supports qui sont de plus en plus nombreux en fonction du nombre de classes et de la variabilité de l'écriture. En conséquence, peu de systèmes utilisent les SVMs comme classifieur [**31**][**63**][**54**][**64**]. Malgré leur complexité, il est toujours intéressant de comparer les performances de tels classifieurs avec celles d'autres classifieurs plus adaptés à des systèmes opérationnels.

Dans ce manuscrit nous allons utiliser ce classifieur d'une part pour comparer ses performances à celles de nos différents classifieurs isolés. D'autre part, il va être utilisé comme un classifieur cible (classifieur de symboles) d'un classifieur hybride que nous proposons dans une approche globale de reconnaissance d'expressions mathématiques manuscrites en-ligne (cf. chapitre 4).

### 2.2.5. Approche d'inférence floue

Les systèmes à base de règles sont initialement utilisés pour les systèmes experts avec leurs expressions sous formes de conditions logiques. Un ensemble de conditions (prémisses) doivent être satisfaites pour valider la conclusion d'une règle. Les systèmes d'inférence floue ont été utilisés avec succès pour la reconnaissance de l'écriture manuscrite depuis les années 90, comme proposé dans [**65**]. Ce dernier classifieur est maintenant utilisé pour la reconnaissance des symboles électriques dans un système de reconnaissance des schémas électriques [**66**]. Les travaux dans [**67**][**68**] proposent également un classifieur de symboles mathématiques à base des règles floues, et aussi pour l'analyse structurelle et syntaxique.

### 2.2.6. Réseaux de neurones artificiels (RNA)

Un réseau de neurones artificiels est formé d'un grand nombre d'unités élémentaires, simples, et fortement interconnectées [**69**]. Le fonctionnement de cette unité de base (appelé neurone) est fondé sur une approximation du fonctionnement du neurone biologique [**70**].

Plusieurs modèles connexionnistes de neurones ont été adaptés pour la reconnaissance de formes et la reconnaissance de l'écriture. Ce sont souvent les éléments de base des systèmes de reconnaissance de formes structurées. George et al.[**71**] ont proposé d'utiliser les réseaux de neurones pour la reconnaissance des symboles du domaine de partitions musicales. Dimitriadis et al. [**72**] ont proposé l'utilisation de réseaux de neurones de type ART (Adaptive Resonance theory). Harteman et al. [**34**], Marzinkewitsch et al.[**73**] ont pour leur part utilisé le fameux PMC (Perceptrons Multi couches). Ce dernier est un classifieur très compétitif et performant qui est souvent utilisé pour la reconnaissance des formes dans différents domaines. Les performances des PMC se rapprochent de celles des SVM, mais ils sont nettement meilleurs en termes de mémoire et de temps de traitement. Par exemple, pour la tâche simple de reconnaissance des 10 chiffres, il faut 452,000 paramètres pour un SVM par rapport à moins de 40,000 paramètres pour un PMC à une couche cachée [**74**]. Par ailleurs, une variante de PMC, les réseaux de neurones à convolution (TDNN), est mise en œuvre pour la reconnaissance de l'écriture avec les mêmes propriétés du PMC en ajoutant une (ou plusieurs) couches de type convolution pour l'extraction de caractéristiques locales du signal d'entrée (cf. chapitre 4). En effet, un TDNN (Time Delayed Neural Network) est insensible aux variations de positions et de translations de l'écriture. Cette structure utilise des informations locales du signal au lieu de n'apprendre que sur une vue globale. L'architecture d'un TDDN se découpe en 2 parties : la partie extraction des caractéristiques pour la ou les couches basses, et ensuite un PMC classique.

Nous adopterons les réseaux de neurones de type PMC et TDNN comme classifieurs au cœur de notre système. Nous sommes surtout attirés par leur capacité à apprendre les exemples d'apprentissage itérativement. Ceci n'est pas possible avec la plupart des autres classifieurs. Nous allons développer davantage les architectures des PMC et TDNN, ainsi que leur algorithme d'apprentissage et leurs optimisations dans le chapitre 4.

### 2.2.7. Segmentation et reconnaissance simultanée

Toutes ces méthodes de classification vues précédemment exigent une segmentation préalable parfaite. Cependant, il existe des méthodes qui effectuent ces deux tâches simultanément. Ainsi, les méthodes basées sur les modèles de Markov caché (HMM pour Hidden Markov Models) permettent de reconnaître directement l'expression (mot, phrase) sans passer par une étape de segmentation. Cette méthode est également efficacement utilisée dans la

reconnaissance de la parole [**75**]. Pour modéliser les symboles, on utilise principalement des HMMs gauche-droite. Les symboles sont modélisés par des HMMs dont le nombre d'états dépend de la taille. Par exemple, 12 états pour les plus grands, 8 états pour les lettres et symboles moyens, et à 3 états pour les plus petits (comme '.' ou ',') [**76**].

Une autre approche consiste à rechercher pendant les étapes de segmentation et reconnaissance, les unités de base de l'expression qui sont les symboles eux-mêmes. On cherche donc à trouver les segments qui sont plus probables pour représenter des symboles. Pour cela on peut se baser sur l'information de reconnaissance pour effectuer la segmentation. En utilisant les techniques de programmation dynamique on propose plusieurs hypothèses de segmentation. On associe à chaque hypothèse un coût venant du score de reconnaissance et des informations structurelles de cette hypothèse. Finalement, on choisit la segmentation ayant un coût minimal. Les hypothèses de segmentation sont des regroupements de traits dans une expression en-ligne [**23**], ou des régions de l'image dans une représentation hors-ligne [**19**].

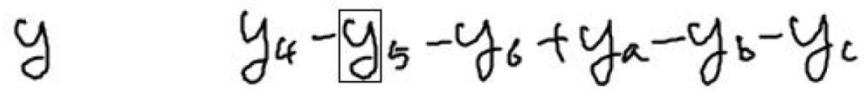
Dans certains graphiques 2D, un trait est partagé entre plusieurs symboles car il est contraignant pour l'utilisateur d'imposer un lever de stylo entre deux symboles successifs. C'est particulièrement le cas des schémas électriques où un seul trait peut représenter plusieurs symboles reliés par des connexions. Dans ce cas, la segmentation se décompose en deux étapes. D'abord il faut trouver des primitives qui représentent des sous parties des symboles, et ensuite regrouper celles appartenant au même symbole. Gennaria et al. ont supposé qu'un symbole ne consiste qu'en segments consécutifs pour réduire l'espace de recherche [**57**], d'autres ont relâché cette contrainte [**16**]. Une première segmentation se fait en se basant sur des informations géométriques sur les traits. Les points des segmentations choisis sont ceux qui portent des changements de comportement du trait. Ces points sont distingués grâce au mouvement observé du stylo, ce sont les points où la vélocité du stylo est minimale, et des changements de direction ou de courbure du trait sont détectés. D'autres points correspondant au point de poser et lever de stylo sont également considérés. Toutes les hypothèses de regroupement des primitives sont ensuite énumérées. Afin de réduire l'espace de recherche, des conditions géométriques sont appliquées pour élaguer les hypothèses peu probables. Finalement, le classifieur de symboles associe une probabilité à chacune des hypothèses et l'ensemble de symboles maximisant la probabilité globale est choisi.

Le système proposé dans [**77**] décompose le trait en un ensemble de primitives de deux types : lignes et arcs. Ces primitives sont extraites par une technique de détection des courbes. Ensuite des heuristiques sont appliquées afin de fusionner certaines primitives d'une façon intelligente (deux lignes horizontales par exemple). Les primitives sont organisées dans une structure

hiérarchique ascendante, où un groupe de primitives forme un composant spécifique, et un groupe de composants forme un symbole. Chaque symbole de cette structure est reconnu par des règles heuristiques basées sur des observations des symboles du domaine considéré. Lorsqu'un symbole est associé à une classe, son facteur de confiance est calculé à partir des informations spatiales des primitives le constituant. Finalement, l'ensemble des symboles ayant le facteur maximal de confiance est le résultat de la reconnaissance.

### 2.2.8. Détection et correction des erreurs

Par ailleurs, plusieurs systèmes proposent une étape de post-traitement afin de corriger les erreurs venant de la segmentation ou de la reconnaissance des symboles. Par exemple, un facteur de confiance obtenu en utilisant une grammaire d'attributs est utilisé dans [**52**]. Ensuite, l'erreur peut être corrigée en choisissant d'autres candidats de la reconnaissance des symboles. Dans un contexte de reconnaissance interactive, il est également envisageable d'inviter l'utilisateur à faire un retour pour résoudre l'erreur potentielle. Chan et al. ont proposé d'étendre la grammaire pour inclure toutes les erreurs prévues dans les productions de la grammaire. Dans ces cas, la grammaire produit en plus des expressions correctes, des situations erronées qui vont nécessiter de corriger l'erreur selon la règle appliquée [**78**]. Lee et al. [**79**] utilisent des connaissances a priori pour corriger quelques types d'erreurs. Pour cela, Ils emploient des règles heuristiques comme par exemple : si on reconnaît $5in$ ou $c0s$ , on considère qu'il s'agit d'erreurs qui sont corrigées automatiquement aux *sin* et *cos*. Par ailleurs, on peut aussi utiliser la consistance du contexte de l'expression reconnue pour corriger quelques erreurs éventuelles [**80**]. On définit trois types de consistance : de forme, de fréquence, et de structure. Comme on le voit à la Figure 27(a), localement il y a une ambiguïté entre 'y' et 'g', mais on choisit d'attribuer le 'y', puisqu'il y a une consistance de répétition de symbole 'y' dans l'expression, Figure 27 (b).



(a) (b)

Figure 27 - Exemple de consistance de répétition

Après cette étape de détection et reconnaissance des symboles par différentes approches, il s'agit maintenant d'interpréter les segments et les symboles trouvés afin de trouver la structure de l'expression.

## 2.3. Interprétation

L'étape d'interprétation vise à déduire de la disposition et de la taille relative des symboles précédemment reconnus une description structurée cohérente de l'expression [**23**]. En fait, l'interprétation d'une expression mathématique consiste à analyser la structure géométrique de l'expression et puis à appliquer une analyse syntaxique. Le but de cette analyse est de trouver l'arbre de dérivation de l'expression qui est ensuite facilement transformable en format standard (LaTeX, MathML …).

### 2.3.1. La représentation de structure et de syntaxe d'expression mathématique

Même s'il existe des standards (comme LateX ou MathML) pour décrire une expression mathématique, ces descriptions ne sont pas directement employées dans les systèmes de reconnaissance d'expressions mathématiques. LaTeX permet de représenter une expression par une chaîne unidimensionnelle, par exemple l'expression '$x^2+y_1$' se décrit par la chaîne *$x^2+y_1$*. Cette chaîne est de plus en plus compliquée quand l'expression se complique. Pourtant, ce type de représentation n'est pas directement utilisable dans une analyse structurelle ou syntaxique. Les standards LaTeX ou MathML sont plutôt utilisés pour l'affichage d'une expression, nous les utiliserons pour décrire la vérité terrain d'une expression mathématique, cf. chapitre 3. En conséquence, on utilise souvent des structures spéciales adaptées à la description des expressions mathématiques. Intuitivement, la nature hiérarchique des expressions mathématiques fait que la plupart des chercheurs ont adopté une représentation sous une forme de graphe (arbre syntaxique ou relationnel). Souvent, la représentation choisie est fortement reliée à la méthode d'analyse. Ainsi, une représentation n'est pas facilement transposable d'une méthode à une autre.

#### 2.3.1.1. Arbre relationnel (SRT : Symbol Relation Tree)

Dans [**24**] l'utilisation des arbres est simplifiée à une description géométrique. Ils introduisent une région cachée d'écriture (Hidden Writing Area HWA) pour chaque nouveau trait. La relation entre ces régions décrit la probabilité de relation spatiale et logique entre les traits. Geneo et al à leur tour utilisent un arbre binaire pour représenter l'expression [**67**]. Les opérateurs occupent les nœuds intérieurs de l'arbre. Ils peuvent être des opérateurs explicites (vus dans l'expression écrite), tels que +, -, *,…etc ; ou implicites comme les indices et les exposants. Ces dernières sont déduits des relations spatiales entres les symboles. Quant aux symboles, ils occupent les feuilles de l'arbre, la Figure 28 montre un exemple de l'arbre qui représente l'expression $28\text{-}x^y$.

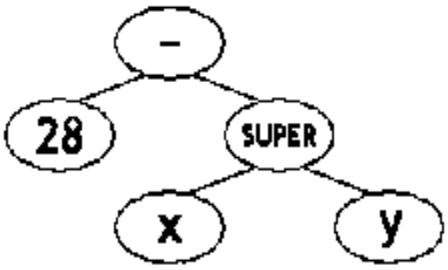



Figure 28 - Exemple d'un arbre binaire d'une expression mathématique

Ce genre d'arbres est connu sous le nom SRT (Symbol Relation Tee), ou l'arbre relationnel des symboles. Un SRT représente simplement les symboles et leurs relations spatiales dans un arbre. Plusieurs variations de SRT ont été proposées depuis les années 90 en gardant toujours le principe de représenter l'expression sous la forme de symboles/relations. La façon de déterminer la relation entre deux symboles varie également selon la méthode d'analyse structurelle, cf. section suivante. Six types de lien sont considérés entre les symboles et les sous-expressions [**41**] (haut, bas, indice, exposant, horizontal, et vertical). Dans la structure qu'ils proposent, un nœud non-terminal est une sous-expression et le type de lien avec ses nœuds fils. Un type de lien spécifique « sous-expression » est nécessaire pour faire le lien avec les nœuds terminaux qui représentent les symboles [**41**][**79**] [**81**].

Une structure similaire est proposée dans [**42**], la racine de l'arbre contient toute l'expression, un nœud intérieur contient une sous-expression et le type de relation entre ses éléments. La relation est une des huit directions : au dessus, en dessous, gauche, droite, au dessus gauche, au dessus droite, en dessous gauche, et en dessous droite, voir Figure 29.

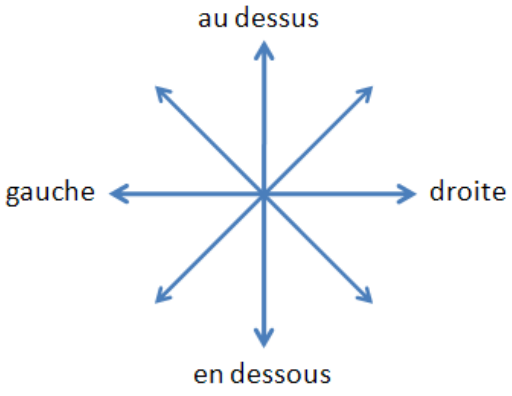



Figure 29 - Illustration des huit directions utilisées dans l'arbre structurel

Dans [**82**] les arcs de l'arbre sont utilisés pour représenter les relations et les nœuds représentent uniquement les symboles. Ce concept est ré-adopté récemment dans plusieurs travaux [**15**] [**83**]. Dans le système proposé dans [**15**] l'arbre est d'abord construit, Figure 30, avec un symbole dominant et ses

sous-expressions comme des fils. Les relations sont encodées dans les arcs. Ensuite, un SRT est construit d'une façon récursive pour chacune des sous-expressions. La relation spatiale est une des suivantes : à l'intérieur, au dessus, en dessous, indice, exposant, et droite. Delaye et al. quant à eux proposent d'utiliser cette structure comme une représentation générique des langages bidimensionnels [**83**]. Leurs expérimentations montrent l'utilité de telle représentation pour les expressions mathématiques, et aussi pour la reconnaissance de caractères chinois.

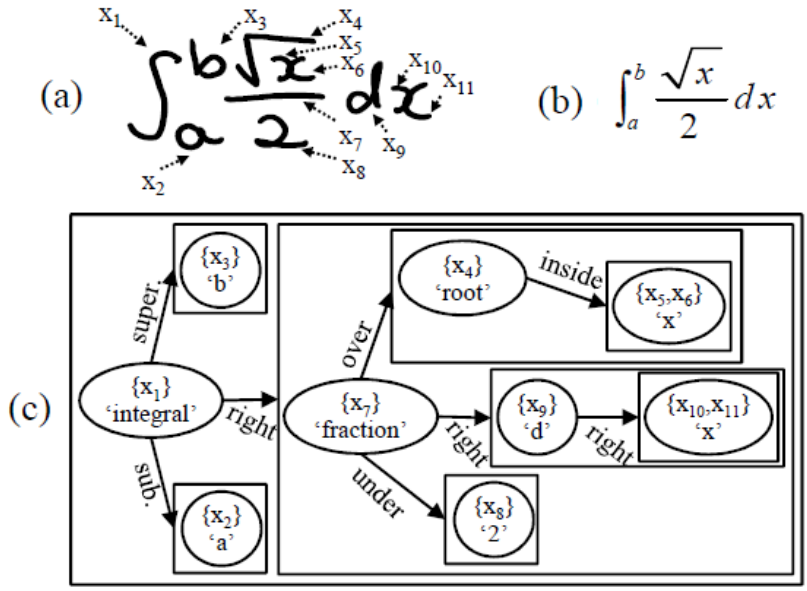



Figure 30 - Exemple d'un SRT, (a) l'expression manuscrite, (b) l'expression prévue, (c) le SRT correspondant [**15**]

### 2.3.1.2. Arbre structurel des lignes de base (BST : Baseline Structure Tree)

Un arbre structurel des lignes de base représente la structure hiérarchique des lignes de base dans une expression [**31**][**53**] [**84**][**85**]. Cet arbre capture l'essentiel de la disposition spatiale des symboles sans prendre le parti d'une interprétation spécifique. L'avantage est qu'on peut représenter une expression par un BST sans qu'elle soit syntaxiquement correcte (ex. *'2+'*), et sans prendre en compte son interprétation sémantique (ex. *'f(x)'* est représentée sans savoir s'il s'agit d'une fonction *'f'* ou d'une multiplication). Un BST contient deux types de nœuds qui se trouvent en alternance dans les niveaux de l'arbre :

- Symbole : ce type de nœud représente un symbole mathématique et contient toutes les informations de ce symbole (identité, type, boîte englobante, …).
- Région : une région représente une partie de l'expression contenant une ligne de base qui représente une relation entre les symboles.

Comme le montre la Figure 31 une région de type « expression » est la racine de l'arbre. Ensuite, tous les symboles qui se situent sur la ligne de base initiale sont considérés comme des nœuds fils de la région expression. Ainsi de suite, toutes les lignes de bases sont expliquées dans l'arbre selon leurs relations avec les symboles du niveau précédent. La relation gauche/droite n'existe pas explicitement, mais l'ordre des nœuds dans le même niveau reflète cet ordre.

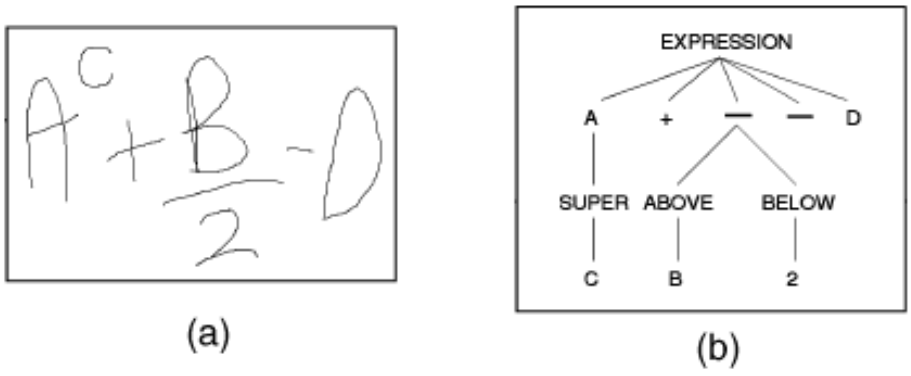



Figure 31 - (a) l'expression manuscrite (b) le BST correspondant [**85**]

### 2.3.2. Analyse structurelle

La structure géométrique d'une expression mathématique est très souvent plus complexe que celle d'un texte. Ainsi, un texte s'écrit quasi systématiquement de gauche à droite alors que les symboles mathématiques peuvent s'écrire dans presque toutes les directions, comme le montre la Figure 32.

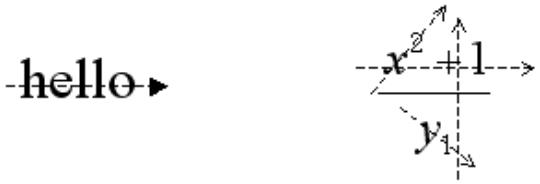



Figure 32 - Différence de directions d'écriture entre un simple texte et une expression mathématique

L'analyse structurelle consiste à extraire des informations spatiales des symboles et à trouver les relations spatiales entre les symboles. Eventuellement le résultat de cette analyse est une description structurelle de l'expression qui représente dans certain cas le résultat final du système. Mais dans la plupart des cas elle sert à alimenter l'analyse syntaxique afin de trouver l'arbre de dérivation de l'expression.

### 2.3.2.1. Les informations spatiales des symboles

Pour bien déterminer le type de relation spatiale entre les symboles, on se base sur des informations structurelles de chaque symbole. Intuitivement, on se base sur la boîte englobante de chaque symbole. Dans ces conditions, on associe à chaque symbole l'ensemble des informations suivantes : ($x_{min}$, $y_{min}$, $x_{max}$, $y_{max}$) [**42**] [**72**], Figure 33(a).

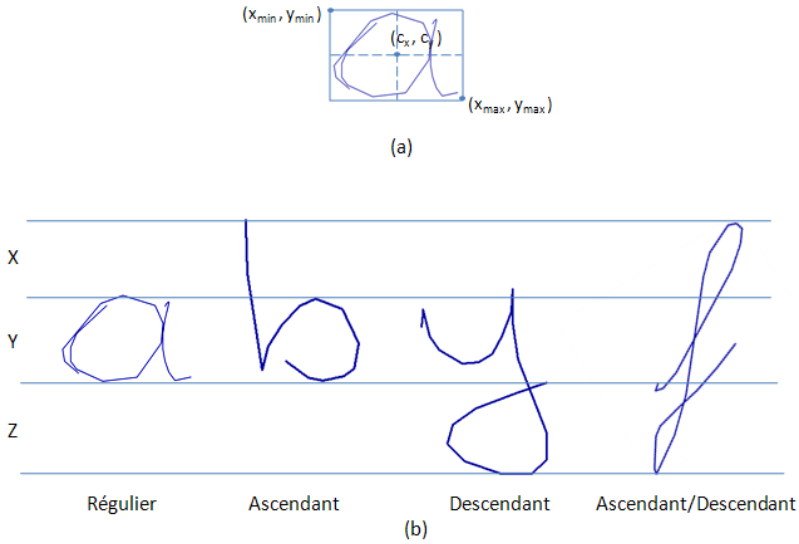



Figure 33 - (a) Informations spatiales d'un symbole, (b) Catégories des symboles

Manifestement, il subsiste des ambiguïtés en ne se basant que sur les boîtes englobantes, cf. chapitre1. Donc, on cherche plutôt à trouver des descriptions structurelles relatives qui dépendent du type des symboles. Comme le montre la Figure 33 (b), on peut distinguer les caractères selon quatre types : régulier, ascendant, descendant, et ascendant/descendant. Le même raisonnement s'applique à l'alphabet grec. D'autres types sont éventuellement considérés afin de couvrir la grande variété des symboles mathématiques (ex : opérateurs, racine carré, symboles élastiques, …).

Zanibi et al. proposent de représenter le symbole par son centre typographique (le centroide)[**85**], où le $x_{centroide}$ est simplement le centre sur l'axe X de la boîte englobante. Alors que le '$y_{centroide}$' est calculé selon le type de symbole. D'une façon similaire, il a été proposé d'utiliser les boîtes englobantes, mais aussi une ligne de base spécifique à chaque symbole selon son type [**76**].

D'autres méthodes évitent l'ambiguïté des boîtes englobantes en normalisant ces boîtes. Ce sont alors les boîtes et les centres normalisés qui sont utilisés pour l'analyse structurelle [**81**][**86**][**87**]. La Figure 33 montre que trois régions X :Y :Z ont été définies pour l'identification du type de symbole selon les valeurs des parties X et Z. Les boîtes englobantes sont ensuite estimées en ajoutant une partie virtuelle descendante ou ascendante (ou les

deux) selon le type de symbole. Toutefois, cette technique ne fonctionne pas quand il s'agit des symboles très petits (ex : ' –' , '.') ou très grands (ex : '[', '|') [**88**].

Il faut également noter qu'il existe quelques symboles irréguliers qui ont des styles différents dans les expressions typographiées selon la police utilisée. Cette irrégularité est beaucoup plus présente dans les expressions manuscrites car les tailles et les styles des symboles varient énormément d'un scripteur à un autre.

#### 2.3.2.2. Relations spatiales

Les relations spatiales entre les symboles sont cruciales pour l'interprétation structurelle de l'expression mathématique. En effet, même si tous les symboles ont correctement été reconnus, il reste le problème de l'interprétation de la structure en deux dimensions de l'expression. Comme nous l'avons vu dans le chapitre 1, les relations spatiales sont de nature floue. Il est très difficile de décider d'une façon absolue si une situation entre deux symboles provient de telle ou telle relation.

Dans l'approche proposée dans [**85**], il est défini des régions imaginaires autour des symboles de la ligne de base initiale, celles-ci dépendent du type de symbole. Dès qu'on détermine le premier symbole de la ligne de base, on ajoute d'une façon récursive tous les symboles ayant des centroides qui tombent dans la région HOR (relation horizontale). Une région définit l'espace prévu pour que les symboles se relient avec la relation associée à cette région, voir Figure 34. En plus, on définit les régions des relations possibles. Par exemple, pour le symbole $\sum$ on prévoit la possibilité de trouver des symboles au dessus, en dessous ou à droite (HOR).

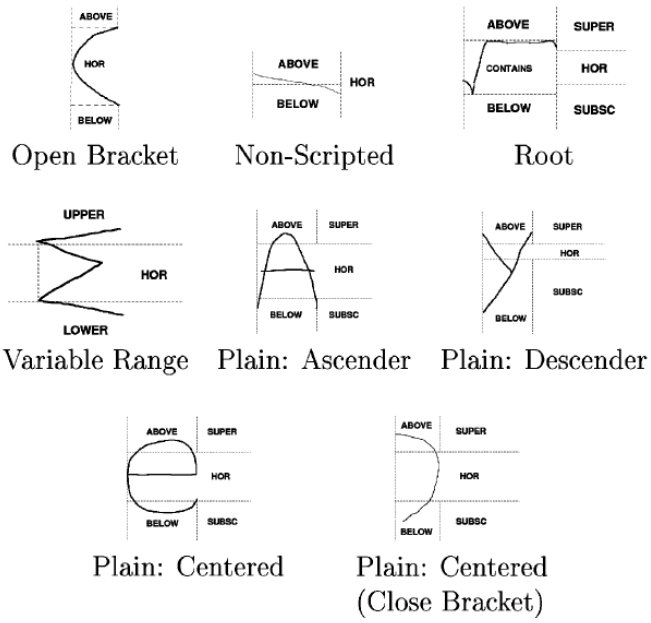



Figure 34 - Régions associées aux différents types de symboles [**85**]

Ce processus est répété récursivement pour chaque région. Il se met en œuvre très aisément grâce à l'utilisation des centroides. De cette façon, on évite des ambiguïtés puisque un centroide tombe forcement dans une région. Mais à cause de la dépendance avec un seul point (le centroide) on risque fortement de se tromper si le centroide est erroné.

Une approche similaire mais en utilisant les boîtes englobantes a été proposée dans [**89**][**90**]. Il est défini autour de chaque symbole 9 régions dans lesquelles un autre symbole est admissible pour qu'il y soit en relation. Ces régions sont définies selon les 8 directions principales (au dessus, au dessous, à gauche, …), auxquelles s'ajoute la notion d'inclusion (à l'intérieur), voir Figure 35. Cette dernière notion est particulièrement importante pour certains symboles comme la racine carrée, par exemple. En effet, cette zone de recherche (région admissible) est proportionnelle à la taille du symbole pour lequel on cherche la relation. Plusieurs critères sont considérés afin de déterminer si une relation existe ou pas entre ces deux symboles. D'abord, on calcule un critère géométrique basé sur l'alignement et la taille des symboles pour chaque région. Un seuil permet de décider quelle relation a la plus forte probabilité. Puis, le type des symboles est pris en compte pour renforcer la décision prise. On définit également des liens interdits pour certains symboles. Par exemple, un opérateur comme < ne peut être un exposant par rapport un autre. Dans la plupart des notations mathématiques $x^{<}$ n'a pas de sens. Une approche similaire est proposée dans [**83**]. Dans cette approche, des modèles géométriques basés sur les boîtes englobantes permettent de calculer un score flou pour chaque relation pour un couple de symboles donnés.

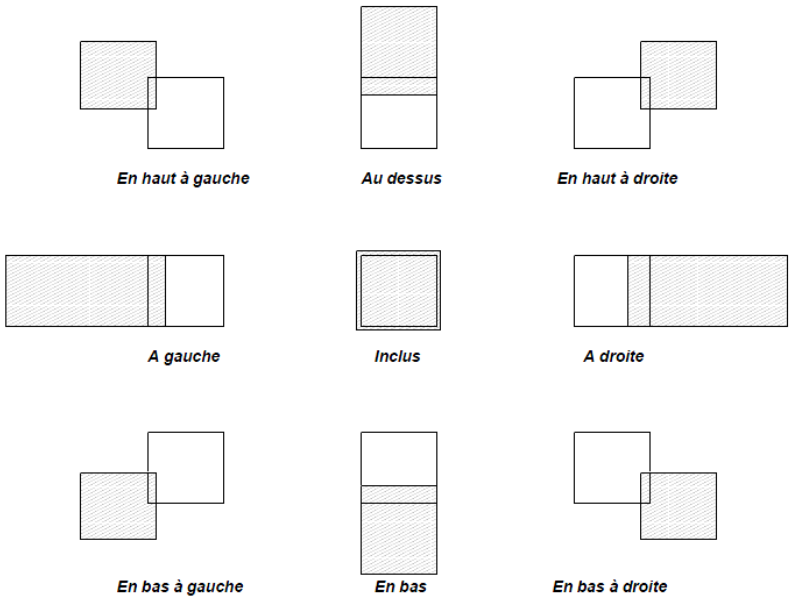



Figure 35 - Définition des régions [**89**]

L'utilisation des règles floues apparaît très appropriée pour définir les relations spatiales lorsqu'il n'est pas possible de décider de façon binaire de la qualité d'une relation. Dans ce cas, la description des relations spatiales est plus naturelle sous la forme d'informations floues. Pour cela, on peut calculer un indice de confiance de chaque relation par une fonction d'appartenance [**91**]. On choisit la configuration globale qui minimise la confiance combinée de toutes les relations. Plusieurs fonctions d'appartenance permettent en quelque sorte de décrire une relation d'une façon proche d'un raisonnement humain. Pour obtenir une telle description, on associe plusieurs fonctions à chaque règle floue [**67**][**68**][**92**]. Prenons l'exemple de la règle *exposant*($s_i$,$s_j$), la fonction multi-confidence fournit les informations suivantes :

- $s_j$ est seulement à droite de $s_i$ ;
- $s_j$ est un peu plus haut que $s_i$ ;
- $s_j$ a beaucoup moins de surface que $s_i$.

Vu le petit nombre de relations spatiales dans les expressions mathématiques, les règles floues sont relativement faciles à configurer et optimiser. Delaye et al. [**9**] ont proposé la notion de paysage flou qui définit la région admissible pour positionner un symbole (ou trait) par rapport un autre. De plus, ces paysages sont appris à partir d'une base d'exemples au lieu d'être définis de façon empirique. Cette méthode a été employée pour la reconnaissance de caractères chinois. En fait, la reconnaissance des caractères chinois comporte les mêmes problématiques que la reconnaissance des expressions mathématiques. En effet un caractère Chinois a une disposition complexe, dans un espace 2D, de traits composés en sous-symboles appelés radicaux.

Aly et al. proposent une méthode basée sur la distribution de certaines caractéristiques des relations spatiales [**88**]. En observant un nombre suffisant d'exemples pour chaque relation, il est possible de construire une carte de distribution. Pour chaque occurrence d'une relation, on déduit le couple (H, D) où H et D sont la taille et la position normalisées de la relation [**87**]. Supposons que $(h_1,c_1)$ et $(h_2,c_2)$ soient les informations spatiales normalisées des deux éléments (symboles, sous-expressions) intervenant dans la relation, on calcule H et D par : $H = \frac{h_1}{h_2} \times 1000$, $D = \frac{c_1 - c_2}{h_1} \times 1000$ ; les valeurs obtenues sont représentées dans une carte de distribution dans l'espace normalisé (H, D), comme celle de la Figure 36.

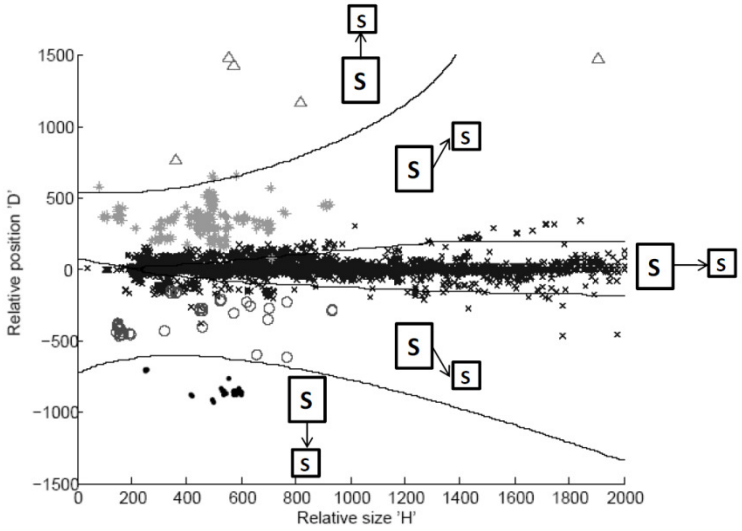



Figure 36 - Exemple de carte de distribution des couples (H,D) normalisés [**88**]

La Figure 36 montre différentes zones dans l'espace représentant les relations. Pour un couple donné, on calcule (H,D) et la relation de la région où se trouve (H,D) est celle la plus probable. Toutefois, ces zones se chevauchent fortement. Les types des symboles sont également utilisés pour obtenir une carte spécifique à chaque type pour faciliter la détermination des relations. La limite de cette méthode est la dépendance des exemples utilisés pour construire les cartes. Mais avec une base qui contient assez d'exemples, on obtient une distribution représentative des relations spatiales.

En fait, les relations spatiales jouent un rôle principal dans l'approche que nous allons proposer. Elles se basent sur les informations spatiales relatives aux types des symboles. Ces informations sont la hauteur et la position de la ligne de base du symbole, ces deux grandeurs dépendant non seulement des paramètres géométriques mesurés mais également du type de symbole manipulé. Les relations que nous proposons portent en plus du sens géométrique, un sens logique (ex : opérateur, somme). Nous proposerons tout

d'abord une approche heuristique pour déduire un coût structurel pour chaque relation possible. Puis, dans un deuxième temps, nous proposerons une approche statistique similaire à celle introduite ci-dessus. Ainsi, on associe à chaque relation un modèle probabiliste qui sert ensuite à déduire le coût de la relation. Notre approche structurelle est détaillée dans le chapitre 4.

### 2.3.3. Analyse syntaxique

La dernière étape de la reconnaissance d'expressions mathématiques consiste le plus souvent en une analyse syntaxique. L'objectif de cette analyse se résume en trois points principaux :

- Produire l'arbre de dérivation final de l'expression dans le but d'obtenir une représentation hiérarchique transformable facilement aux standards comme LaTeX ou MathML ;
- Assurer l'exactitude grammaticale de l'expression reconnue ;
- Et, ce qui est plus important : déduire un contexte global de l'expression afin d'enlever des ambiguïtés rencontrées tout au long de la reconnaissance.

Une expression mathématique est générée par un langage formel bidimensionnel. Une grammaire hors contexte peut parfaitement produire toutes les expressions mathématiques. Il suffira donc de disposer d'un analyseur (parseur) efficace pour obtenir l'arbre de dérivation de l'expression. Les grammaires hors contexte sont efficacement employées pour les langages formels 1D (comme les langages de programmation). La grammaire 2D peut être construite directement à partir des expressions mathématiques d'apprentissage [**18**]. Remarquons qu'il faut beaucoup d'expressions représentatives pour pouvoir généraliser la grammaire résultante. Plus récemment, une extension en grammaire 2D hors contexte a été proposée pour modéliser les relations entre les symboles mathématiques [**19**][**93**]. On définit une région par un rectangle dans l'image d'entrée. Supposons $N \to A \oplus B$ soit une règle de production de la grammaire. Elle dénote la production de la région $N$ en unifiant les régions $A$ et $B$. La règle est appliquée si une certaine contrainte est respectée, de plus un coût est associé à cette production.

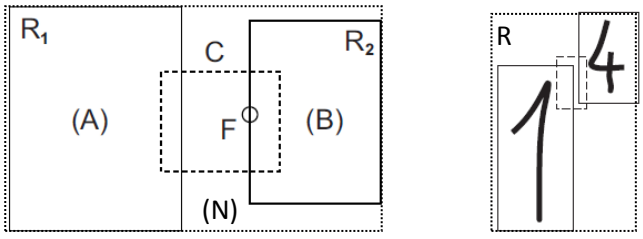



Figure 37 - Schéma général de production de la règle $N \to A \oplus B$

La contrainte à respecter est montrée dans la Figure 37, les régions $R_1$,$R_2$ (A,B) sont unifiées si un point caractéristique de $R_2$ (le point F) est à l'intérieur

du rectangle C relative au type de règle de production. La région résultante R est le rectangle le plus petit contenant $R_1$ et $R_2$. La pénalité est calculée en fonction des tailles et positions relatives de A et B. Les règles de productions sont appliquées successivement jusqu'à unifier toutes les régions en une seule qui représente l'expression.

Cependant, analyser les langages bidimensionnels n'est pas directe et nécessite des algorithmes spéciaux et des contraintes pour diminuer la complexité [**21**]. On trouve dans la littérature deux approches principales pour effectuer l'analyse syntaxique : une analyse basée sur une grammaire, ou bien une autre se basant sur un graphe.

#### 2.3.3.1. Approche grammaticale

Chan et al. proposent une méthode basée sur une « Definite Clause Grammar » (DCG) [**59**]. Tout d'abord, l'expression est transformée de sa forme bidimensionnelle en une représentation unidimensionnelle. En effet, il existe beaucoup de parseurs 1D efficaces. La DCG est utilisée pour définir un jeu de règle de remplacement pour analyser les expressions. Ces règles s'écrivent facilement sous forme de prédicats Prolog, mais cette approche n'est pas très efficace car elle implique un grand nombre de retours en arrière. Pour contourner cette difficulté, les auteurs ont proposé d'améliorer l'efficacité du système en transformant les règles pour qu'elles soient factorisées à gauche « left factored ». Cette dernière méthode permet une analyse descendante et améliore nettement la performance de la DCG, notamment en nombre d'opérations logiques appliquées. Plus récemment, Garain et al. proposent une méthode se basant sur une grammaire hors contexte [**94**]. En s'appuyant sur des informations géométriques extraites de l'expression, ils effectuent l'analyse à l'aide de la grammaire. Initialement, la structure de l'expression est construite en divisant l'expression récursivement en bandes horizontales et verticales jusqu'à ce qu'un niveau atomique soit atteint (un seul symbole). Finalement, chaque élément atomique est fusionné en suivant les règles de production correspondant aux relations spatiales, afin d'obtenir l'équivalent Latex. Dans [**24**], il est proposé une grammaire probabiliste et structurelle. Chaque règle de la grammaire est reliée à une relation logique (au dessus, en dessous, …). Une probabilité est aussi associée à chaque règle. Donc, la reconnaissance de l'expression est transformée en une recherche des règles maximisant la probabilité d'obtenir une certaine interprétation.

Une approche appelée « Fuzzy Shift Reduce Parsing » (FSRP) est présentée dans [**92**]. Cette méthode est construite sur une analyse ascendante traditionnelle, donc assure une vérification syntaxique efficace. La logique floue est introduite pour faire face aux ambiguïtés structurelles inhérentes à l'écriture manuscrite. Ils proposent un nouvel algorithme, le « Fuzzy Online Structural Analysis algorithm » (FOSA). L'arbre de l'expression est

immédiatement mis à jour pour chaque nouveau trait de l'utilisateur. La logique floue est utilisée à chaque étape de la reconnaissance pour pallier aux imprécisions de l'écriture manuscrite [**67**].

Zanibbi et al. proposent pour leur part une technique basée sur la transformation d'arbre [**85**]. Ainsi, une recherche récursive permet d'identifier les structures linéaires de l'expression et l'on construit un arbre structurel des lignes de base « BST » (voir §2.3.1.2). Le BST est ensuite soumis à une analyse lexicale pour transformer les relations spatiales en relations logiques. Cet arbre est traduit en chaîne de caractères et ensuite analysé avec une grammaire hors contexte classique (comme celle utilisée pour les langages de programmation) pour produire l'arbre syntaxique de l'expression.

Dans l'approche que nous allons proposer (cf. chapitre 4), nous allons simplifier la complexité de l'utilisation d'une grammaire bidimensionnelle. Pour cela, nous proposons une grammaire qui comporte deux types de règles : horizontales et verticales. Un coût structurel est associé à l'application de chaque règle en fonction des types de symboles et des relations spatiales mises en jeu.

#### 2.3.3.2. Approche graphique

Une grammaire graphique s'applique sur un graphe sous la forme d'un système de réécriture avec analyse descendante. La réécriture consiste à remplacer un sous graphe par un seul nœud contenant l'arbre syntaxique de l'expression. Les grammaires de graphes sont exploitées dans [**95**] pour la reconnaissance de schémas électriques et d'organigrammes. L'unité de base est le segment, où l'ensemble des segments du graphique sont organisés en forme du graphe en fonction de leur proximité. La grammaire permet ensuite de transformer ce graphe en un autre en fusionnant les segments en symboles ou connexions

Dans une grammaire de graphe pour interpréter les expressions mathématiques, les parties droites des règles de production sont localisées dans le graphe et représentées dans un seul nœud [**34**][**76**][**96**]. Toutefois le temps du calcul est considérablement élevé. L'algorithme de parsing est simple, on applique itérativement la première règle valide jusqu'il ne reste plus des règles applicables. Malheureusement deux règles peuvent réécrire un même graphe de deux manières différentes et donc obtenir deux arbres différents. Pour éviter des ambiguïtés on propose d'introduire du contexte pour les règles. On considère deux critères : la priorité mathématique de l'opérateur décrit dans la règle, et une information graphique. Ce dernier critère est important pour déterminer la priorité à droite, par exemple l'exposant à une priorité plus grande que l'opérateur '+'[**76**] [**89**].

## 2.4. Optimisation simultanée de la segmentation, la reconnaissance et l'interprétation

Cette approche est de plus en plus adoptée ces derniers temps. En effet, s'il s'agit toujours d'utiliser les différentes approches proposées ou adaptées auparavant, ces différentes étapes se fondent dans un cadre global et s'appliquent simultanément pour reconnaître l'expression. L'idée s'inspire simplement de la perception humaine de l'expression mathématique. En fait, un humain perçoit la totalité de l'expression sans faire des perceptions et traitements séquentiels. D'où l'idée de simuler la perception humaine en faisant les tâches de segmentation, reconnaissance et interprétation de façon concomitante. L'expression la plus probable sera issue directement de la coopération des trois modules : la segmentation, la reconnaissance des symboles et l'interprétation. Un grand avantage de cette approche est de limiter la propagation des erreurs héritées d'une étape à une autre et de permettre de récupérer des solutions localement moins bonnes mais conduisant globalement à une meilleure solution.

Yamamoto et al. proposent de modéliser tout le processus de reconnaissance d'une expression manuscrite en-ligne par une grammaire stochastique hors contexte [**24**]. La grammaire prend en compte l'ordre de l'écriture et la nature 2D des symboles. Elle contient également des règles de production des traits et des symboles. Chacune de ces règles est probabiliste, i.e. à l'application d'une règle on calcule sa probabilité en utilisant des méthodes classiques de classification de symboles. Le reste des règles modélise les relations spatiales et la syntaxe de l'expression en calculant une probabilité structurelle associée à chacune de ces règles. La probabilité structurelle est calculée en fonction des boîtes englobantes et de la région cachée de l'écriture. Une hypothèse d'une expression $X$ est donc dérivée de la grammaire $G$, où $X=\{p_1,p_2,...,p_N,q_1,q_2,...,q_M\}$ ; $p_n =< A_n \rightarrow B_n C_n, S_n >$ est la règle qui génère la sous-expression $\mathrm{A_n}$ de $\mathrm{B_n}$ et $\mathrm{C_n}$ sous la condition structurelle $\mathrm{S_n}$ et $\mathrm{q_m}$ génère les traits et les symboles. On cherche donc à optimiser la probabilité de génération de l'expression $X$ qui s'écrit : $P(X) = \underset{X \in E_x}{arg\,max} \prod_{n=1}^{N} P(p_n) \prod_{m=1}^{M} P(q_m)$ ; où $P(p_n)$ et $P(q_m)$ dénotent la probabilité des règles. Le problème de la recherche de l'expression la plus probable dérivée de la grammaire est résolu en utilisant l'algorithme CYK (Cock-Younger-Kasami)[2], (voir Figure 38).

Malgré l'efficacité de l'algorithme CYK, celui-ci repose sur une contrainte forte : il exige le bon ordre temporel des traits qui composent l'expression. Son utilisation nécessite donc soit d'imposer à l'utilisateur de respecter la

[2] http://en.wikipedia.org/wiki/CYK_algorithm

chronologie des traits, soit de mettre en place un traitement supplémentaire pour bien ordonner les traits en cas de présence de traits tardifs.

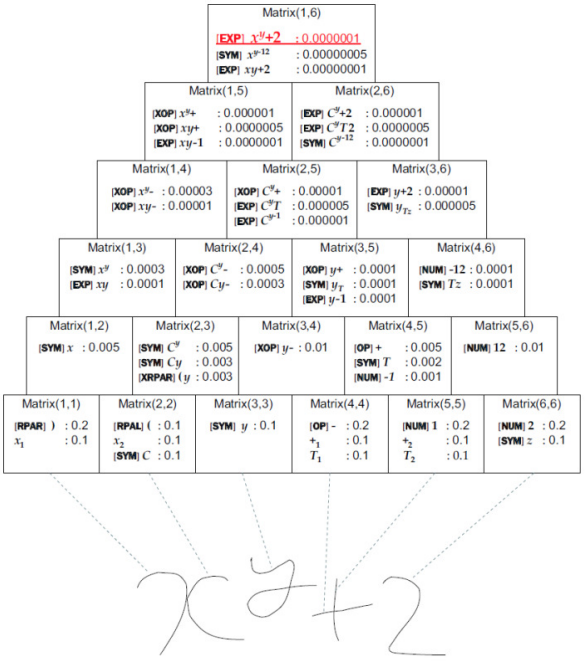



Figure 38 - Exemple de recherche de l'expression la plus probable en utilisant l'algorithme CYK

Afin de pallier le problème d'ordonnancement temporel des traits, Rhee et al proposent une structure de recherche à couche (layer search framework) pour la reconnaissance d'expressions manuscrites en-ligne [**15**]. La reconnaissance d'expressions est donc reformulée comme un problème de recherche de l'interprétation la plus probable d'un ensemble donné de traits d'entrée. On cherche alors les symboles les plus probables qui sont reliés à leur tour par les relations spatiales les plus probables. La structure de l'expression est étendue en ajoutant des hypothèses de symboles les unes après les autres, représentant les différentes identités des symboles et à chaque ambiguïté, une nouvelle branche est rajoutée. La recherche est initialisée par une structure vide, ensuite elle est étendue en ajoutant les hypothèses des symboles qui utilisent un sous-ensemble des traits qui restent.

Ce processus crée un arbre de recherche où chaque nœud représente les structures des expressions déjà construites avec leurs coûts estimés des scores de classification des symboles et des coûts de relations spatiales entre eux. Cette recherche s'effectue par l'algorithme de recherche « best-first ». On applique à chaque candidat des règles simples de validité contextuelle pour résoudre des ambiguïtés locales (ex : fermeture de parenthèse). Un coût heuristique d'admissibilité est calculé à chaque nœud, il mesure la probabilité que les traits qui restent peuvent conduire à une solution acceptable. Par

conséquent, les branches non-admissibles sont élaguées, réduisant ainsi l'espace de recherche. La condition d'admissibilité assure que la première solution qui comprend tous les traits est celle la plus probable.

Si les deux approches globales ci-dessus se focalisent sur la reconnaissance d'expressions mathématiques, la méthode DALI proposée dans les travaux de thèse de Sébastien Macé [**26**] essaye d'étendre l'approche globale à une modélisation générique pour la reconnaissance des langages 2D et à l'utilisation d'une reconnaissance à la volée. Cette méthode se base sur une notion de contexte structurel modélisé à l'aide d'une nouvelle classe de langages visuels : les grammaires de multi-ensembles à contraintes pilotées par le contexte (GMC-PC). Ce formalisme permet de modéliser un document structuré (graphique 2D) et ses symboles tout en restant sensible au contexte. Bien entendu, l'exploitation du contexte au sein des grammaires va augmenter la complexité de l'analyse. Le but est de cibler des interprétations cohérentes d'un élément en fonction d'autres déjà existants. Ainsi, les processus de segmentation, de reconnaissance et d'interprétation vont interagir de façon à limiter les hypothèses à considérer, comme illustré dans la Figure 39.

Une propriété intéressante de cette méthode est de permettre de choisir les contraintes à imposer à l'utilisateur afin d'augmenter la performance d'analyse et diminuer l'espace de recherche. Ces contraintes sont modélisées à l'aide du formalisme grammatical lié à la notion du contexte structurel. En effet, les contraintes se présentent en imposant l'ordre d'introduction des symboles. De plus, le concepteur peut s'appuyer sur les conventions de composition du domaine du langage 2D, autrement dit, de la manière dont les utilisateurs composent traditionnellement ces symboles. Par exemple, le fait qu'un disjoncteur soit toujours dessiné après l'interrupteur auquel il est associé.

La méthode DALI se différencie des précédentes par la propriété d'une analyse à la volée, avec une interprétation incrémentale avec retour visuel. Donc à chaque application de production d'une règle de la grammaire, les traits participants sont remplacés par leur équivalent reconnu. De plus, une production de symboles peut se décomposer en plusieurs règles, ce qui rend possible des retours visuels intermédiaires du symbole. Ce retour visuel progressif va permettre à l'utilisateur de devenir un acteur du processus d'analyse. Donc, l'utilisateur peut continuer la composition s'il est satisfait par le retour de ce qu'il vient d'entrer, ou bien corriger des erreurs éventuelles.

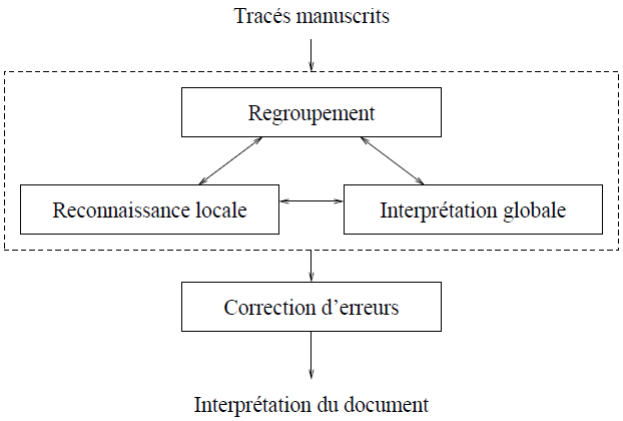



Figure 39 - Structure générale de la méthode DALI [**26**]

Une limitation majeure de cette méthode est que les GMC-PC nécessitent une intervention humaine forte pour l'écriture des règles correspondantes au langage concerné. L'ajout de pré-conditions et de post-condition rajoute du travail supplémentaire surtout qu'il y a beaucoup de règles à écrire. Puisqu'il faut parfois modéliser différentes règles de composition pour un même symbole.

D'autres exemples d'approches globales proposées pour la reconnaissance d'expressions mathématiques se trouvent dans [**19**][**97**]. Il est à remarquer que quelque soit l'approche globale proposée, le problème est généralement reformulé en une optimisation d'une fonction globale.

C'est sur ce même principe d'approche globale que nous allons construire l'architecture de notre système. Nous transférons le problème de reconnaissance en recherche de la meilleure interprétation possible d'un ensemble de traits d'entrée, cf. chapitre 4. La reconnaissance d'expressions est faite a postériori, c'est-à-dire que l'expression est saisie en entier avant d'être reconnue. L'architecture du système proposé va permettre de compenser certaines limitations des méthodes existantes. Elle permettra également d'entraîner le classifieur de symboles et les relations spatiales directement des expressions mathématiques comme nous allons le voir dans le chapitre 4, ce que nous appelons un ***apprentissage global***.

Peu de travaux ont présenté la traduction de l'arbre de dérivation en format standard. Quelques exemples de possibles transformations de l'arbre syntaxique en format de LaTeX ou celle de Gnuplot sont montrés dans [**98**]. En fait, il suffit de parcourir l'arbre syntaxique afin d'obtenir la traduction désirée.

## 2.5. Vers un reconnaisseur générique des langages 2D : Reconnaissance des schémas électriques manuscrits en-ligne

Nos travaux sur la reconnaissance d'expressions mathématiques prennent la suite de ceux effectués sur la reconnaissance des schémas électriques. Ces travaux ont été menés au sein de la thèse de Guihuan Feng [**1**]. Elle a proposé une méthode pour la reconnaissance et la segmentation simultanées des schémas électriques afin de localiser les symboles électriques et les connexions. Le système se base sur la même brique industrielle fournie par « Vision Objects » : le générateur d'hypothèses (cf. chapitre 4). Néanmoins, aucune information contextuelle n'était utilisée. En conséquence, on n'obtient en résultat que des symboles sans avoir aucune interprétation logique du schéma.

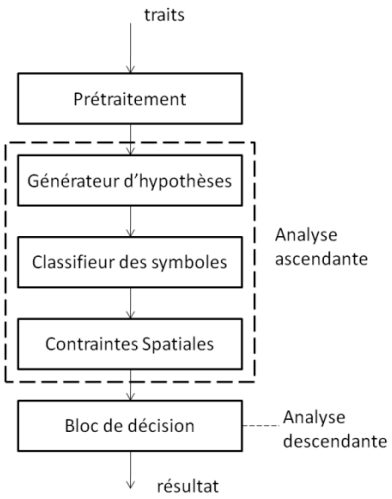



Figure 40 - Architecture globale du système de reconnaissance des schémas électriques

La Figure 40 montre l'architecture du système. Contrairement aux expressions mathématiques, un seul trait peut représenter plusieurs symboles. Donc, le prétraitement consiste à trouver les primitives (segments). Des segments qui représentent des lignes ou des arcs sont identifiés grâce à une modélisation par HMM (Hidden Markov Model). Ensuite, le générateur d'hypothèses énumère des regroupements possibles de segments en respectant quelques contraintes spatiales (la distance, intersection, …). D'autres contraintes sont indispensables afin d'éviter d'exploser l'espace de recherche : le nombre maximal d'hypothèses et le nombre maximal de traits par symbole. Un point très important du générateur est qu'il se base sur une programmation dynamique 2D permettant le regroupement des segments non-consécutifs. Cette propriété augmente la liberté de dessiner un schéma en permettant de dessiner des traits tardifs.

Des contraintes de connectivité sont utilisées afin d'améliorer la performance du système. Dès que le classifieur associe une hypothèse à une classe de symbole, on lui attribue un score intégrant les autres symboles qui lui sont connectés selon son type. Par exemple, si on reconnaît une résistance horizontale, on cherche la présence de connecteurs aux extrémités gauche et droite. Il y a donc un coût structurel et un de reconnaissance qui sont combinés pour toutes les hypothèses. Finalement on cherche l'ensemble de regroupements de symboles minimisant le coût global combiné de toutes les hypothèses.

Nous avons retenu les observations suivantes :

- Malgré les résultats prometteurs obtenus, la base de donnée utilisée pour l'évaluation est relativement petite ce qui peut limiter ces résultats.
- Pendant la phase de génération d'hypothèses, on obtient des hypothèses invalides qui ne représentent pas de vrais symboles. Le comportement du classifieur face à ces hypothèses peut dégrader les performances du système.
- Les contraintes de connectivité présentent en quelque sorte une modélisation contextuelle. Néanmoins, pour obtenir une interprétation complète du graphique à reconnaître, un modèle de langage basé sur une grammaire est indispensable.

Dans les travaux menés, nous allons essayer de compenser ces limitations. Nous présenterons tout d'abord une méthode de génération de grosses bases d'expressions mathématiques manuscrites en-ligne. Un schéma d'apprentissage global est proposé pour faire face aux hypothèses invalides. Finalement une modélisation contextuelle est proposée afin d'effectuer une analyse structurelle et syntaxique pour obtenir des interprétations complètes plutôt que se limiter à localiser les symboles.

## 2.6. Conclusion

Nous avons exploré dans ce chapitre les différentes techniques et les approches proposées depuis les années 70. On a vu que la reconnaissance d'expressions mathématiques (en-ligne/hors-ligne) ou (typographiées/manuscrites) se décompose classiquement en quatre sous problèmes : la segmentation, la classification des symboles, l'analyse structurelle et l'analyse syntaxique. Ce schéma de reconnaissance a été adapté et optimisé en employant des méthodes inspirées de plusieurs domaines, notamment, la reconnaissance de formes, et l'analyse syntaxique et structurelle.

Chacun des systèmes proposés a ses avantages et inconvénients. Toutefois, la comparaison des performances entre les différentes approches n'est pas directement possible pour plusieurs raisons. Nous allons dans le chapitre suivant présenter ce problème d'évaluation d'un système de reconnaissance d'expressions mathématiques, en abordant les problèmes de base de données, de représentation de vérité terrain et de mesures de performance.

# CHAPITRE 3 : LE PROBLÈME DE L'ÉVALUATION

La reconnaissance d'expressions mathématiques suscite beaucoup d'attention ces dernières années. Si un certain nombre de systèmes apparaissent dotés de capacités de traitement intéressantes, l'évaluation objective de ces systèmes reste sujette à de nombreuses difficultés. En effet, il y a une lacune de méthode universelle pour évaluer leurs performances. En conséquence, d'un point de vue global, la comparaison des résultats obtenus est limitée pour plusieurs raisons. Premièrement, peu de base d'expressions d'écriture en-ligne/hors-ligne ne sont aujourd'hui publiquement accessibles. La disponibilité d'une telle base est pourtant une étape indispensable pour le développement, la mise au point et l'évaluation d'un système de reconnaissance d'expressions. Les chercheurs tendent donc à collecter leurs propres bases d'expressions. Les bases se limitent à des sous-classes d'expressions ou à certains domaines. Deuxièmement, chaque système adopte sa propre structure de données pour représenter les expressions, comme nous l'avons vu dans le chapitre 2. Ce qui fait qu'il n'existe pas une façon standardisée pour décrire la vérité terrain d'une expression mathématique. De plus, il n'y a pas de mesures communes d'évaluation d'un système de reconnaissance d'expressions mathématiques. Dans ces conditions, il n'est pas possible de faire une comparaison directe des performances de ces systèmes.

Nous allons dans ce chapitre explorer quelques bases et mesures d'évaluation existantes. Nous développons davantage la méthodologie que nous avons adoptée pour obtenir nos bases de données et évaluer notre système de reconnaissance d'expressions mathématiques manuscrites en-ligne.

## 3.1. La base de données d'évaluation : Quoi ? Combien ? Comment ?

La création d'une base de données d'expressions est une étape indispensable pour permettre de concevoir, de développer et de tester un système de reconnaissance d'expressions mathématiques. En effet, elle servira à évaluer le taux de reconnaissance et à valider les différentes spécifications du système (bases d'évaluations et de tests). Parmi les points importants figure le choix du corpus utilisé pour bâtir cette base de données. A l'inverse des bases de données purement textuelles, peu de références sont disponibles dans le domaine des expressions mathématiques.

Il y a deux stratégies pour obtenir une base d'expressions. La première stratégie consiste à choisir un corpus d'un petit nombre d'expressions ($E$) et on collecte la même expression plusieurs fois ($N$) pour avoir une base d'une taille de $E \times N$ expressions. En général, on considère un corpus d'expressions couvrant certains domaines. Quand il s'agit d'une base d'expressions en-ligne, la grande difficulté est de trouver des participants pour saisir les expressions pour augmenter la variabilité de styles d'écriture. Lehmberg et al. ont choisi un corpus de 27 expressions écrites près de cinq fois par 12 scripteurs pour un total de 1538 expressions [**56**]. Fukuda et al. se limitent à 4

expressions mais contenant toutes les structures traitées par leur système [**23**]. Vingt scripteurs ont écrit chaque expression deux fois pour un ensemble de 160 expressions. Il est bien évident qu'une telle base ne pourra pas refléter la capacité du système, puisqu'il y a des situations imprévisibles d'expressions. Une solution est de couvrir plusieurs domaines des mathématiques. Par exemple, quatre domaines ont été choisis dans [**33**] : l'algèbre élémentaire, les fonctions trigonométrique, la géométrie analytique, et les intégrales indéfinies. Un ensemble de 60 expressions est écrit par 10 scripteurs pour un total de 600 expressions.

D'une façon complémentaire, la deuxième stratégie consiste à choisir un nombre relativement élevé d'expressions. Ensuite, chaque scripteur participe à ne saisir qu'un sous-ensemble de la totalité des expressions. Par exemple dans [**68**] huit scripteurs participent à constituer une base de 245 expressions manuscrites toutes différentes.

Habituellement, les scripteurs sont choisis dans les milieux scientifiques pour que les expressions soient les plus naturelles possibles. Rhee et al. [**15**] collectent leurs bases d'expressions parmi des étudiants en études supérieures. Ils ont constitué deux corpus nommés : KME-I et KME-II composées de 30 et 100 expressions respectivement, voir le Tableau 1. On constate que la tâche de collecter une base d'expressions mathématiques manuscrites n'est pas facile. Car alors un grand nombre de scripteurs est requis pour obtenir une base bien représentative.

En ce qui concerne le domaine hors-ligne, des expressions typographiées sont disponibles dans des documents scientifiques, il suffit alors de trouver les bons documents (livres, journaux, manuels, cahiers d'examen, etc.) pour créer une base hors-ligne. L'avantage est que l'on obtient facilement des pages complètes contenant des expressions ce qui peut être utile pour l'évaluation d'autres systèmes (extraction d'expressions, analyse de documents) [**20**]. Mais, un effort supplémentaire est nécessaire pour isoler les expressions des documents. Okamoto et al. [**99**] ont extrait 2842 expressions de 2255 pages des journaux mathématiques au cours d'un projet d'archivage. Les images de documents sont simplement numérisées par un scanneur en résolutions différentes (300dpi, 600dpi, …) selon le prétraitement effectué par le système. La base UW-III (University of Washington English/Technical Document Database III) [**100**] est publiquement disponible. Mais elle ne contient que 100 images d'expressions qui ne semblent pas être suffisamment représentatives. Garain et al. [**101**] proposent un corpus pour la recherche sur les expressions mathématiques. Ce corpus est très large et comporte 274 classes de symboles distincts. L'originalité de ce corpus est d'utiliser des documents synthétisés en plus de documents scannés pour augmenter la base. Un nombre total de 2459 expressions isolées est extrait dont 62 expressions proviennent de la base Aster initialement conçue pour la synthèse vocale d'expressions mathématiques

[**102**]. Ce corpus a été étendu à 70 expressions qui ont été saisies par 20 scripteurs deux fois pour une base de 2800 expressions manuscrites [**103**].

Dans la plupart des cas le modèle de langage du système est, en quelque sorte, adapté au corpus choisi. Alors que l'inverse est plus réaliste. Remarquons que l'on apprend aux enfants (puis étudiants) à écrire de nouvelles expressions et pas simplement un ensemble d'expressions. Une grammaire entraînable a été présentée dans [**18**], mais cette voie n'a pas été explorée davantage dans les années suivantes. Plus récemment, MacLean et al. [**104**] élaborèrent l'idée de décrire un corpus par une grammaire. Vingt étudiants ont été recrutés pour saisir les expressions. Trente trois expressions extraites des livres de lycée sont rédigées par tous les scripteurs une fois résultant en 660 expressions. De plus, des expressions générées aléatoirement de la grammaire ont été saisies par les scripteurs pour augmenter le nombre d'expressions à 4655 expressions manuscrites.

Par ailleurs, l'étiquetage des expressions est long et fastidieux, mais cette étape est indispensable pour l'évaluation. Nous discuterons des différents niveaux d'étiquetage envisageables et des outils d'étiquetages nécessaires pour mieux évaluer un système de reconnaissance dans la dernière partie de ce chapitre, cf. §3.5. De toute façon, la performance d'un système dépend directement du corpus d'expressions, et la complexité de celui-ci peut varier dans des très grandes proportions.

Le Tableau 1 montre une comparaison sommaire entre les bases d'expressions recensées. Nous pouvons constater que ces bases sont très différentes aux niveaux de leur taille, complexité, variabilité, et représentativité, ce qui rend impossible la comparaison de systèmes évalués indépendamment sur ces différentes bases.

Avoir une base d'expressions de taille conséquente permet de tester sur un corpus particulier un système. Mais pour concevoir un système performant, il peut être intéressant de moduler la complexité de la base, ce qui est très coûteux si des expressions réellement écrites sont utilisées. Nous avons donc proposé un outil qui permet de générer une base d'expressions manuscrites de taille importante à partir de n'importe quel corpus d'expressions mathématiques à l'aide d'une base de symboles isolés.

Tableau 1 - Comparaison des bases d'expressions mathématiques manuscrites

| | #Expressions du corpus | #Scripteurs | #Expressions de la Base | #Moyen de symboles par expression | #Classes de symboles |
|---|---|---|---|---|---|
| **(Lehmberg et al. 1996)** [**56**] | 27 | 12 | 1538 | 27 | - |
| **(Fukuda et al. 1999)** [**23**] | 4 | 20 | 160 | 24 | - |
| **(Chan et al. 2001)** [**33**] | 60 | 10 | 600 | - | - |
| **(Fitzgerald et al. 2007)** [**68**] | 245 | 8 | 245 | 15-20 | - |
| **(Prusa et al. 2007)** [**19**] | 330 | - | 330 | - | - |
| **(Genoe et al. 2006)** [**67**] | 60 | 2 | 60 | 12 | - |
| **(Yamamoto et al. 2006)** [**24**] | 15 | 50 | 336 | - | 52 |
| **KME-I (Rhee et al. 2009)** [**15**] | 30 | 13 | 390 | 13 | 50 |
| **KME-II (Rhee et al. 2009)** [**15**] | 100 | 31 | 3100 | 14 | 79 |
| **(Garain et al. 2003)** [**103**] | 70 | 20 | 2800 | 9 | 142 |
| **(Shi et al. 2007)** [**97**] | 2574 | - | 2574 | 17 | 150 |
| **(Maclean et al. 2010)** [**104**] | 4655 | 20 | 4655 | 5 | - |

## 3.2. Un générateur d'expressions mathématiques manuscrites en-ligne (LaTeX2Ink)

LaTeX2Ink permet de générer des expressions pseudo-synthétiques en arrangeant des symboles isolés d'une façon stochastique guidée par la chaîne LaTeX de l'expression désirée. Bien qu'il soit moins intéressant de tester un système avec des données artificielles, cet outil reste utile pour générer facilement une nouvelle base d'expressions manuscrites d'un corpus donné. De plus, l'originalité de notre approche globale impose le besoin d'un grand nombre d'expressions qui permettront d'entraîner les différents algorithmes retenus (base d'apprentissage). Ce générateur s'avèrera donc très utile dans notre méthodologie.

L'objectif final du système de reconnaissance est de pouvoir fournir la chaîne LaTeX équivalente de l'expression fournie en entrée sous la forme d'encre numérique. Ici, le processus est inversé : le générateur doit fournir à

partir de la chaîne LaTeX représentant l'expression, un fichier correspondant à l'écriture manuscrite en-ligne de l'expression. La création du fichier s'effectue en quatre étapes montrées dans la Figure 41.

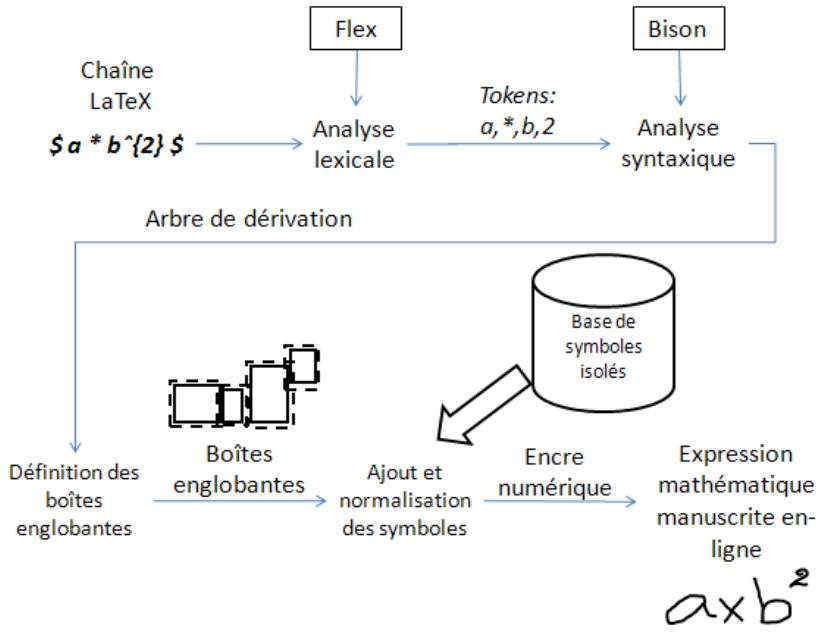



Figure 41 - Structure générale du générateur d'expressions mathématiques

### 3.2.1. Structure du générateur

La première des phases est l'analyse syntaxique de la chaîne LaTeX. Elle est effectuée à l'aide de l'utilisation conjointe de deux outils : Flex et Bison.

***Flex***

Flex est l'implémentation GNU de l'analyseur lexical Lex. C'est un outil pour générer des « scanners » (programmes qui reconnaissent des motifs lexicaux dans du texte). A partir de fichiers d'entrée décrivant le scanner à générer, flex génère un fichier source en langage C, pour produire un programme exécutable. Quand celui-ci est lancé, il analyse le texte en entrée afin d'y trouver des occurrences correspondant aux précédentes expressions régulières « les tokens ».

***Bison***

Bison est un générateur d'analyseur syntaxique qui convertit une grammaire hors-contexte en un analyseur de type LALR [**105**] (Look-Ahead Left Recursive) pour cette grammaire. Il permet de développer des analyseurs pour de multiples langages, du plus simple au plus complexe. La grammaire utilisée pour la génération d'expression se trouve dans l'annexe A-2.

#### 3.2.1.1. Analyse lexicale de la chaîne LaTeX

Flex effectue l'analyse lexicale de la chaîne LaTeX en entrée. Il renvoie alors un lexème (ex : RL : Roman Letter, AN : Arabic Number, FRAC :

Fraction, AS : Flèche …) correspondant à la classe reconnue, une liste complète de classes définies se trouve dans l'annexe A-1. Ensuite, Bison récupère les « tokens » (c'est-à-dire la classe de l'unité lexicale) et effectue une analyse ascendante.

#### 3.2.1.2. Analyse syntaxique : création de l'arbre de dérivation

L'analyse syntaxique effectuée par Bison, permet de construire l'arbre de dérivation de la chaîne analysée. Chaque lexème trouvé va être représenté par un nœud de l'arbre de dérivation. Il peut présenter un nœud terminal, ou bien non-terminal avec un fils gauche et un fils droit. L'ajout des nœuds au cours de l'analyse ascendante permet de récupérer l'arbre complet de dérivation lors de l'analyse syntaxique. Celle-ci est basée sur une grammaire hors contexte. La Figure 42 montre l'exemple d'un arbre de dérivation obtenu à partir de la chaîne LaTeX d'une expression.

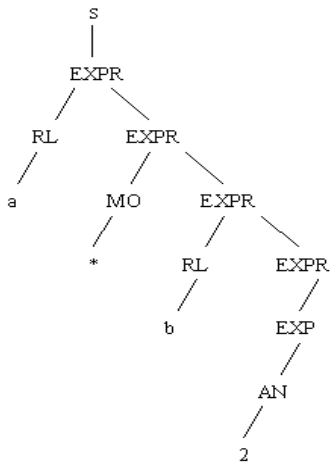



Figure 42 - Arbre de dérivation pour la chaîne : $a*b^{2}$

#### 3.2.1.3. Définition des boîtes englobantes

Une fois l'arbre de dérivation de la chaîne LaTeX obtenu, on effectue un parcours en « profondeur d'abord » afin de définir le placement et la taille des boîtes englobantes associées à chaque élément. Cette étape va nous permettre de définir la structure géométrique de l'expression. Par exemple la rencontre d'un nœud exposant va faire varier l'ordonnée et la taille de ses éléments fils. Pour définir les espacements et les tailles des éléments de l'expression mathématique nous nous sommes basés sur les normes utilisées pour les éditeurs[3] mathématiques, Tableau 2.

[3] Valeurs par défaut des éditeurs classiques (MS équation, Open Office, etc.)

Tableau 2 - Normes éditeurs d'expressions mathématiques

| | |
|---|---|
| Line spacing | 150% |
| Matrix row spacing | 120% |
| Matrix column spacing | 100% |
| Superscript height | 45% |
| Subscript depth | 25 |
| Sub/superscript gap | 15% |
| Limit height | 25% |
| Limit depth | 100% |
| Limit line spacing | 100% |
| Numerator height | 35% |
| Denominator depth | 100% |
| Fraction bar overhang | 1 pt |
| Fraction bar thickness | 0,5 pt |
| Sub-fraction bar thickness | 0,25 pt |
| Slash/diagonal fraction gap | 8% |
| Fence overhang | 1 pt |
| Horizontal fence gap | 10% |

| | |
|---|---|
| Operator spacing (% of normal) | 100% |
| Non-operator spacing | 100% |
| Character with adjustment | 0% |
| Minimum gap | 8% |
| Radical gap (vertical) | 17% |
| Radical gap (horizontal) | 8% |
| Radical width (% of normal) | 100% |
| Embellishment gap | 1,5 pt |
| Prime height | 45% |
| Box stroke thickness | 5% |
| Strike-through thickness | 5% |
| Matrix partition line thickness | 5% |
| Radical stroke thickness | 5% |

| | |
|---|---|
| Full | 10 pt |
| Subscript/Superscript | 80% |
| Subscript/Superscript | 70% |
| Symbol | 170% |
| Sub-symbol | 120% |

Cette étape va permettre de définir la structure géométrique de l'expression, où l'on définit une boîte englobante pour chaque élément terminal. La boîte est définit par les quatre variables suivantes : bbx, bby (abscisse et ordonnée de la boîte), bbw, bbh (la largeur et la hauteur de la boîte).

Lorsqu'on rencontre un élément terminal on sélectionne alors le symbole correspondant pour le scripteur choisi dans la base des symboles isolés. A partir de l'échantillon manuscrit du symbole, on normalise ses coordonnées à l'aide de la hauteur définie plus tôt dans le parcours de l'arbre. Quatre tailles de base ont été définies selon le type de symbole rencontré :

- taille élément ascendant et descendant (*f*) : l'unité de base U ;
- taille élément ascendant ou descendant (*g h* ...etc.) : 5/7 U ;
- taille élément de base (*a* + ...etc.) : 3/7 U ;
- taille petit élément (. , …etc.) : 1/7 U.

La taille d'un élément varie bien entendu en fonction de son rôle dans l'expression. Ainsi, un élément en exposant, en indice, un argument d'intégrale, d'une somme ou d'une limite ont des tailles spécifiques.

Certains éléments d'une formule mathématique ont un comportement élastique et doivent être traités de manière particulière. Ainsi pour les éléments

comme la barre de fraction et la barre de la racine carrée, il est nécessaire de connaître la taille des éléments fils et de remonter cette information afin d'adapter la longueur de cette barre aux éléments qu'elle recouvre. Le même problème se pose pour la hauteur des différents délimiteurs comme les parenthèses, les crochets, etc.

Une fois que la position et la taille nominales de chaque symbole sont fixées, des perturbations aléatoires sont introduites. L'objectif est de permettre des réalisations différentes à partir d'un même ensemble de symboles pour une équation donnée. Pour cela, un bruit Gaussien centré est ajouté sur les variables définissant la position (abscisse et ordonnée) et le facteur d'échelle de chaque symbole.

Il sera ensuite aisé de vérifier la robustesse d'un reconnaisseur vis-à-vis du niveau de bruit appliqué sur le générateur. On pourrait envisager également facilement à ce niveau de rajouter d'autres perturbations telles qu'un angle de rotation, ou un facteur de cisaillement. Cela permettrait d'étendre les capacités de ce générateur stochastique d'expressions mathématiques.

#### 3.2.1.4. Ajout des symboles dans les boîtes englobantes

A la fin de la troisième étape on obtient un arbre avec toutes les valeurs de boîtes englobantes renseignées. Alors, pour chaque élément terminal de l'arbre on récupère les traits du symbole correspondant pour le scripteur retenu. Une fois l'ensemble de l'arbre parcouru on obtient alors une suite de traits correspondant à l'expression manuscrite de la chaîne LaTeX de départ. Le grand avantage du générateur est de pouvoir étiqueter l'expression au niveau de chaque trait au moment de sa génération. Une étiquette correspondant au nom du symbole ajouté est associée à chaque trait. De plus, nous associons un indice unique aux traits d'un même symbole afin d'éviter l'ambiguïté en cas de plusieurs occurrences de ce symbole, un exemple de cet étiquetage est montré dans la Figure 49 (page : 97).

### 3.2.2. Protocoles de générations d'une base d'expressions

L'objectif de la base générée est d'avoir une grande quantité d'expressions pour l'apprentissage et l'évaluation du système. Bien évidemment, il y a des contraintes à respecter au moment de la génération de la base tout en gardant sa diversité. Dans notre approche globale, la base d'apprentissage ne sert pas seulement à régler le système, mais aussi à apprendre le classifieur et le modèle contextuel. Nous retenons deux simples conditions à suivre dans la base d'apprentissage générée :

- tous les symboles reconnaissables par le classifieur doivent être présents.
- toutes les structures reconnaissables par le système doivent être présentes.

Bien entendu, un seul exemple de chaque structure ou symbole n'est pas souhaitable. En fait, il nous faut un corpus d'expressions à générer pour construire la base d'expressions à partir d'une base de symboles isolés.

#### 3.2.2.1. Un corpus d'expressions statiques/dynamiques

Le scénario le plus simple est de générer des expressions à partir d'une liste de chaînes LaTeX. La liste doit contenir des expressions qui représentent le mieux le domaine concerné par le système. Au minimum il faut que la liste respecte les deux conditions ci-dessus pour obtenir une base de couverture minimale. Il est évident que plus les expressions différentes sont nombreuses, meilleur sera l'apprentissage. Dans la base de test on doit s'attendre à n'importe quel jeu d'expressions. Nous allons appeler ce type de liste : un ***corpus statique***. Supposons un reconnaisseur d'expressions qui accepte le vocabulaire {*a b c + =*} et les opérations d'addition et d'exposant. Pour entraîner ce reconnaisseur, nous pouvons utiliser la liste statique suivante qui comporte cinq expressions :

$$\{\ a+b=c,\ a^b+c^b,\ c+b+a,\ a^a+b=c+c+b,\ a+b+c=b+c+b\ \}$$

La taille de la base (*T(b)*) dépendra du nombre de scripteurs participants (*S*). Nous aurons donc à la fin de la génération les mêmes expressions (*E*) du corpus écrites *S* fois : $T(b)=E\times S$.

Pour augmenter la variabilité de la base nous pouvons étendre le scénario précédant en utilisant un ***corpus semi-dynamique***. Il s'agit toujours d'une liste d'expressions mais elle va servir en tant que modèle. Pour chaque expression générée la même structure est gardée et on fait varier les symboles d'une façon aléatoire. En conséquence, on obtient cinq expressions différentes pour chaque scripteur, comme par exemple :

$S_1$:$\{\ b+b=c,\ c^b+b^b,\ a+c+b,\ a^c+b=c+a+b,\ a+c+b=c+a+b\}$

$S_2$:$\{\ a+b=a,\ b^b+a^c,\ a+b+b,a^a+c=a+a+a,\ a+b+c=b+a+c\}$

$S_3$:$\{\ c+b=c,\ a^c+c^a,\ b+a+c,\ c^a+b=c+b+a,\ a+b+b=b+a+a\}$

$S_n$: ….

La liste semi-dynamique permet également facilement d'élargir le vocabulaire dans la base d'expressions. Il suffit de permettre d'ajouter des symboles d'une même classe dans les modèles d'expressions. Comme par exemple ajouter l'opération de multiplication '*x*' et de division '÷' en complément du plus '+' ; et ajouter d'autres lettres en complément de *a*, *b*, et *c*.

Bien entendu dans les deux scénarios précédents la performance du système dépend de sa capacité à généraliser ce qu'il apprend. Il faut qu'il soit capable

par exemple de reconnaître l'expression : $a^{b+b+g+c+a} + m^{f+d+s+b} = x$ *ou* encore $a^{b^c}$ même si elles n'ont jamais été rencontrées lors de l'apprentissage.

Pour augmenter encore plus la variabilité de la base au niveau symboles et aussi au niveau structures, on peut se baser sur un ***corpus dynamique*** d'expressions générées à partir d'un modèle de langage (une grammaire), comme cela est présentée dans [**104**]. Le modèle doit être adapté au domaine concerné par le reconnaisseur. Par exemple, on peut utiliser la grammaire suivante pour générer une base d'expressions qui contiennent des opérateurs binaires et des exposants pour le vocabulaire { a..*z, +, x, ÷, − , =*} :

Exp → subExp = supExp

subExp → subExp Op subExp | $Var^{subExp}$ | Var

Var → a..z

Op → +, *x*, ÷, −

Après avoir déterminé le corpus d'expressions à générer il faut choisir les symboles réels de la base de symboles isolés qui seront utilisés. Chaque scripteur a saisi tous les symboles de la base isolé. Nous pouvons donc utiliser ses échantillons pour générer les expressions manuscrites.

#### 3.2.2.2. Une base d'expressions mono-scripteurs vs. avec scripteurs virtuels

En réalité, une expression est normalement écrite par un seul scripteur. De même, la méthode classique adoptée pour générer les expressions est le mode ***mono-scripteur***. Les symboles utilisés pour une expression sont donc tirés de la base isolée d'un seul scripteur.

L'inconvénient principal des expressions mono-scripteurs est que le même échantillon du symbole isolé est utilisé dans toutes les expressions de ce scripteur, car chaque symbole isolé n'a été saisi qu'une seule fois. C'est surtout gênant quand le même symbole se répète dans plusieurs expressions. Par exemple le *'c'* dans la liste $S_1$ ci-dessus se présente huit fois pour chaque scripteur. Alors nous aurons huit échantillons identiques à la taille près, ce qui ne correspond pas à des expressions écrites globalement par un scripteur réel. Comme le montre la Figure 43-(D), les symboles qui se répètent (*2 a b*) dans une expression réelle sont différents à chaque occurrence. Par contre, ils sont identiques dans les deux expressions synthétiques, Figure 43-(A, B). Pour éviter cela on pourrait introduire des déformations aléatoires à chaque occurrence dans l'expression générée.

Nous proposons une deuxième solution en introduisant ce qu'on appelle un ***scripteur virtuel.*** L'idée est de renforcer la base de symboles isolés en augmentant le nombre d'échantillons disponibles de chaque symbole, surtout ceux les plus fréquents. Ensuite, on génère la base de *E* expressions autant de

fois que le nombre de scripteurs virtuels. Pour cela, pour chaque expression on choisit les échantillons utilisés aléatoirement dans la base des symboles isolés sans imposer la contrainte d'unicité du scripteur. Donc dans une expression générée, les occurrences du même symbole seront différentes, voir la Figure 43-(C). Il est à remarquer que le nombre d'échantillons disponibles dans la base isolée doit être plus élevé que celui de scripteurs virtuels afin d'assurer une bonne diversité d'occurrence des symboles les plus fréquents.

Il est difficile de mesurer la qualité objective de telles productions à défaut de disposer de critères d'évaluation. Nous nous sommes limités à visualiser les résultats obtenus et à les comparer avec les expressions correspondantes écrites globalement par un scripteur. Nous pourrons bien sûr également comparer les performances obtenues par notre système de reconnaissance sur ces bases artificielles de grandes tailles avec des bases réelles, elles, de plus petites tailles.

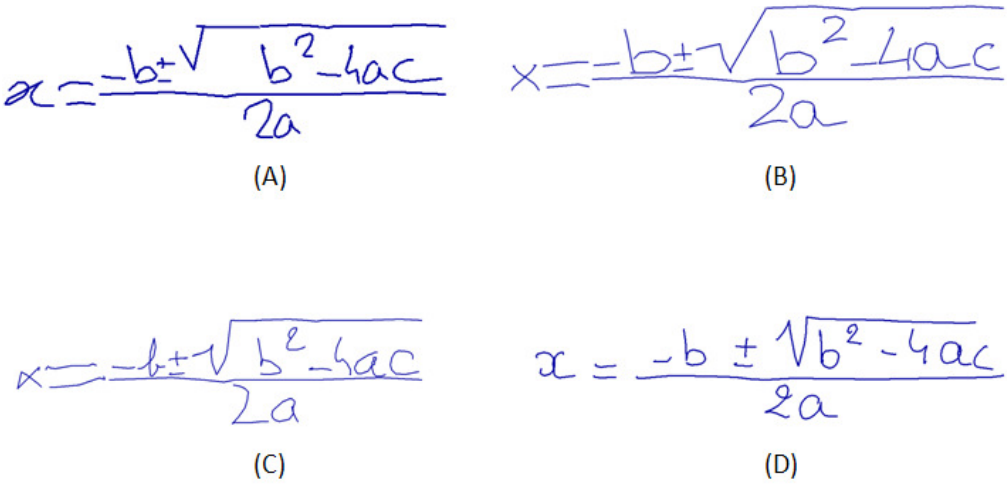



Figure 43 - (A),(B) Expressions générées (mono-scripteurs) (C) Expression générée (scripteur virtuel) (D) Expression réelle

Dans la Figure 43, les expressions (A) et (B) sont des résultats du générateur provenant de deux scripteurs différents en mode mono-scripteur, (C) est un résultat d'une expression générée en mode scripteur virtuel. Par contre, (D) présente une expression manuscrite écrite globalement par un seul scripteur. D'un point de vue visuel, nous sommes capables de générer avec un rendu réaliste n'importe quelle expression désirée afin d'enrichir la base des expressions.

Néanmoins, le générateur réalisé ne vise pas à remplacer les expressions réelles, son but est de fournir une large quantité d'exemples afin d'être capable de tester des situations et des expressions quelconques. Plus précisément, il permet d'obtenir une large base d'apprentissage, ce qui est indispensable pour l'approche globale que nous allons proposer. Cela ne dispensera pas ensuite de tester le système de reconnaissance avec des expressions réelles.

## 3.3. Définition des corpus d'expressions

La première étape fût de rechercher un ensemble d'expressions mathématiques les plus diversifiées possibles couvrant les différents domaines utilisant des formules mathématiques. Nous commençons par présenter les corpus existants Garain et Aster dont nous nous sommes inspirés pour définir trois corpus de niveaux de difficultés croissants.

### 3.3.1. Les corpus existants « Garain » et « Aster »

Nous avons considéré les travaux effectués par Garain [**106**], nous avons récupéré l'ensemble du corpus qui a été utilisé pour tester le système qu'il a proposé qui était destiné à la reconnaissance d'expressions imprimées. Celui-ci est constitué d'expressions provenant de livres scientifiques d'universités écrits en anglais (la plupart de mathématiques) afin de couvrir un large domaine d'expressions. Ce corpus définit 100 expressions s'inscrivant dans les domaines présentés dans le Tableau 3.

Tableau 3 - Corpus base Garain

| Domaines | #Expressions |
|---|---|
| Algèbre | 22 |
| Calcul | 10 |
| Equations différentielles | 10 |
| Intégrales | 10 |
| Logique & théorie des ensemble | 8 |
| Statistiques/ probabilités | 10 |
| Trigonométrie/ géométrie | 10 |
| Vecteur | 10 |
| Misc. (physiques, etc.) | 10 |
| Total | 100 |

Ce corpus comporte 116 symboles distincts et 31 symboles en moyenne par expression.

Le corpus de la base ***Aster***, tiré des travaux de Raman [**102**] regroupe lui aussi un ensemble de 62 d'expressions intéressantes dans différents domaines (cf. Tableau 4) ; certaines équations sont tirées du livre de Knuth, qui les avait utilisées pour montrer le pouvoir de rendu de LaTeX. Il comporte 50 symboles distincts et 13 symboles en moyenne par expression.

Tableau 4 - Corpus base Aster

| Domaines | # expressions |
|---|---|
| Fractions | 8 |
| Exemples de Knuth | 7 |
| Fraction continue | 1 |
| Expressions algébriques | 3 |
| Séries | 5 |
| Logarithmes | 4 |
| Intégrales | 6 |
| Racines | 3 |
| Sommes | 3 |
| Exposants/Indices | 6 |
| Limites | 2 |
| Trigonométries | 7 |
| Fonctions hyperboliques | 3 |
| Distance | 1 |
| Expression quantifiée | 1 |
| Puissances | 2 |
| Total | 62 |

### 3.3.2. Le corpus « Calculette »

Les expressions de ce corpus sont réduites à une simple application de calculatrice. Le nombre de classes de symboles est 15. Ces symboles sont les chiffres [0-9], les simples opérations mathématiques [+, −, x, ÷] et le signe égal [=]. Les expressions ont été aléatoirement générées à partir du modèle « [1-999] [+, –, ×, ÷] [1-999] = [1-999]» :

Exp → Operand Op Operand = Operand
Op → +| – | × | ÷
Operand → [1..9] (70%) | [10..99] (20%) | [100..99] (10%)

Remarquons que la partie droite de l'équation n'est pas forcement le résultat de l'opération pour augmenter la variabilité.

### 3.3.3. Le corpus « RamanReduced »

En raison de la grammaire que nous avons implémentée, nous avons considéré un sous-ensemble de 36 expressions de la base Aster couvrant une bonne partie des disciplines communes des mathématiques montrées dans le Tableau 5. Le nombre total de symboles du corpus est de 412 symboles. Pour ce sous-ensemble, le nombre des classes de symboles distincts est réduit à 34 classes et 11.5 symboles par expression, cf. annexe B-3 pour la fréquence des symboles dans le corpus. Nous appelons ce corpus ***RamanReduced***, il va servir comme corpus de base pour l'apprentissage et évaluation de notre système. Les 36 expressions sont montrées dans l'annexe B-2.

Tableau 5 - Le corpus RamanReduced extrait de la base Aster

| Domaines | # Expressions |
|---|---|
| Fractions | 8 |
| Exemples de Knuth | 6 |
| Fraction continue | 1 |
| Expressions algébriques | 3 |
| Racines | 2 |
| Trigonométries | 6 |
| Logarithmes | 3 |
| Séries | 3 |
| Intégrales | 1 |
| Sommes | 2 |
| Fonctions hyperboliques | 1 |
| Total | 36 |

Tableau 6 - Les 34 symboles du corpus RamanReduced

| | |
|---|---|
| Chiffres | 1..5 |
| Lettres latines | a b c d e i k n x v |
| Lettres grecques | $\delta \pi \vartheta$ |
| Opération binaires | $+ - = \pm \neq$ |
| Symboles élastiques | $\Sigma \int \sqrt{\ } ( ) - \ldots$ |
| Fonctions | log sin cos sinh cosh |

Les 34 classes des symboles comprennent des chiffres, des lettres latines, des lettres grecques, des opérations binaires, des symboles élastiques et des fonctions. Nous considérons le signe moins '–' et la barre de fraction dans la même classe dans notre classifieur, c'est à l'interprétation de faire la différence entre les deux.

### 3.3.4. Le corpus Wiki_CIEL

Avec ce corpus, l'intention est de disposer d'une large base ouverte d'expressions. Pour réaliser cette tâche, nous pouvons définir une grammaire et un ensemble de symboles et générer un corpus dynamique, support de cette base. Pour se rapprocher plus de la réalité et garantir la variété des expressions dans la base, il est possible d'utiliser des expressions qui se trouvent dans des documents scientifiques réels. Wikipédia représente une source très importante de documents scientifiques. Ainsi, nous avons recensé près de 77 000 expressions en format LaTeX dans 7 000 pages web de Wikipédia français. Ensuite, nous pouvons appliquer des filtres pour extraire des sous-ensembles de ce corpus pour les utiliser dans nos expérimentations. Le filtrage peut se faire en se basant sur les domaines scientifiques, la longueur des expressions (nombre de symboles), un ensemble de symboles, des contraintes structurelles, etc.

Nous allons dans la suite détailler nos bases d'expressions utilisant les corpus ci-dessus.

## 3.4. Les bases de symboles/expressions mathématiques manuscrites

Dans nos travaux nous avons utilisé deux types d'expressions : les expressions synthétiques générées avec l'outil LaTeX2Ink présenté précédemment et les expressions réelles complètement écrites par un scripteur. Pour utiliser LaTeX2Ink, nous devons aussi disposer d'un ensemble de symboles isolés. Nous commençons donc dans cette partie par présenter nos bases de symboles isolés, puis les bases que nous avons synthétisées et enfin nos données réelles.

### 3.4.1. Symboles isolés

#### 3.4.1.1. La base CIEL

Nous avons défini un ensemble de 223 symboles mathématiques les plus fréquemment utilisés et permettant de reconstituer l'ensemble des corpus d'expressions. Ces symboles se répartissent suivant les catégories définies dans le Tableau 7. L'ensemble des symboles est détaillé dans l'annexe B-1.

Tableau 7 - Catégories et nombres de classes de symboles mahtématiques

| Catégorie | Nombre |
|---|---|
| Chiffres | 10 |
| Lettres romaines minuscules | 26 |
| Lettres romaines majuscules | 26 |
| Lettres grecques | 30 |
| Flèches | 14 |
| Opérateurs binaires | 45 |
| Symboles élastiques | 10 |
| Délimiteurs | 7 |
| Symboles divers | 29 |
| Fonctions standards | 26 |
| Total | 223 |

L'ensemble des symboles a été récolté à l'aide de la technologie « Anoto » de type stylo/papier digital mise en œuvre par notre partenaire industriel « Vision Objects ». Les stylos sont munis d'une caméra qui décode une zone correspondante à une grille de 6 x 6 du papier, cette grille comporte un motif défini par des points qui sont imprimés à 4 positions possibles (haut, bas, gauche, droite) par rapport à l'intersection de base. On a donc $4^{36}$ motifs différents, chaque motif correspondant à une position précise dans un repère cartésien couvrant une très grande surface. Ces stylos sont donc capables de connaître leurs positions à tout moment. Chaque échantillon est sauvegardé sous le format Unipen [**107**]. Ce format est le fruit d'un compromis entre quarante entreprises réunies au sein d'un consortium. Il fût défini afin de

disposer d'un format standard pour comparer les différents travaux dans le domaine. La Figure 44 montre un exemple d'un fichier Unipen récolté pour le symbole 'a' (un seul trait). Un exemple de symbole à deux traits '+' est également montré dans la Figure 45.

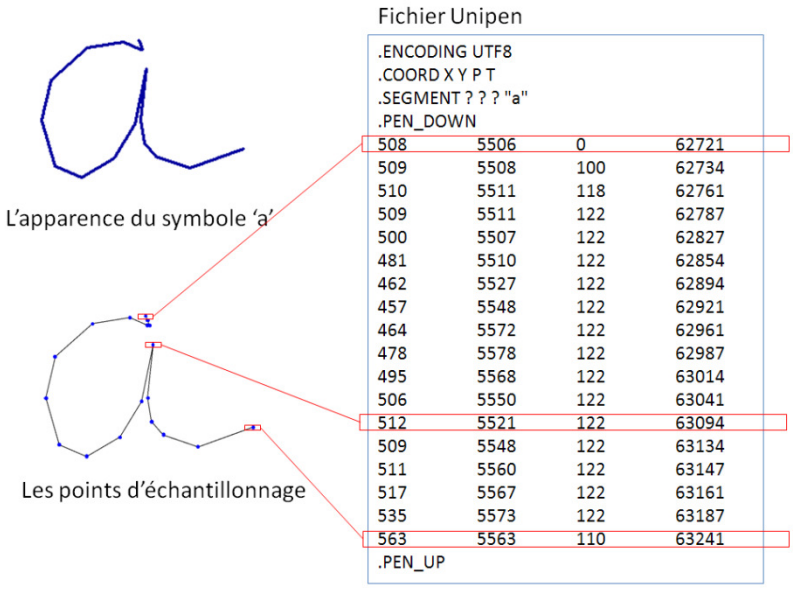



Figure 44 - Le fichier Unipen correspondant au symbole 'a'

La première ligne donne le format du fichier, ici UTF8. La deuxième ligne caractérise les données numériques (la première d'entre elles correspond à l'abscisse, la deuxième à l'ordonnée, la troisième à la pression et enfin la quatrième au temps. La troisième ligne donne l'étiquette du symbole, ici « a ». Ensuite, .PEN_DOWN correspond à un poser de stylo, .PEN_UP à un lever et enfin les données numériques qui sont échantillonnés à différents intervalles de temps.

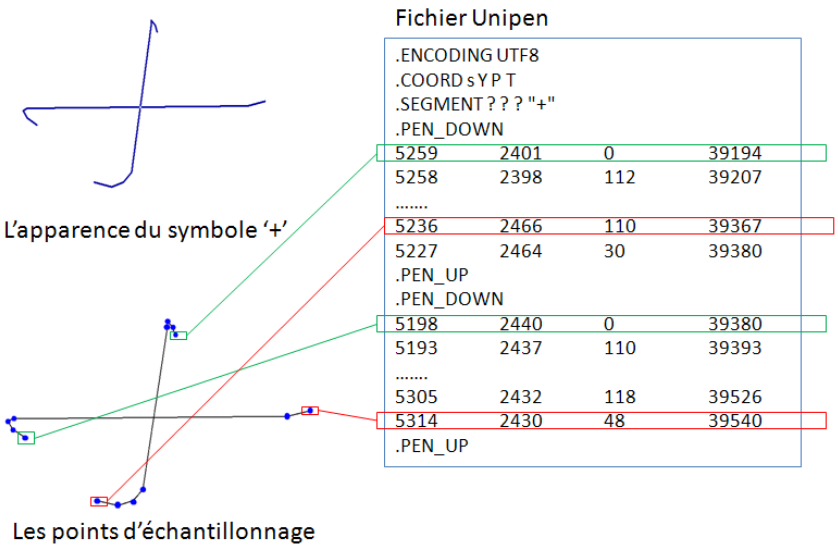



Figure 45 - Le fichier Unipen correspondant au symbole '+'

Après récolte, la base de données est constituée de 62 440 échantillons de symboles isolés produits par 280 scripteurs. Nous allons séparer cette base en deux sous-ensemble, voir Tableau 8, constitué l'un de 180 scripteurs pour la base d'apprentissage, et le reste de 100 scripteurs pour la base de test.

Tableau 8 - Constitution de la base CIEL isolée

| Base | #Scripteurs | #Symboles |
|---|---|---|
| **Apprentissage** | 180 | 180 x 223 = 40140 |
| **Test** | 100 | 100 x 223 = 22300 |

A partir de cet ensemble d'échantillons il est maintenant possible de générer les bases nécessaires d'expressions manuscrites en-ligne.

#### 3.4.1.2. La base IRONOFF

La base de données IRONOFF (IRESTE ON/OFF database) est une base de données duales en-ligne et hors-ligne, collectée et distribuée par notre équipe de recherche. Elle contient un nombre important de caractères isolés, de chiffres et de mots en français et anglais au format UNIPEN. Cette base de données a été créée de telle sorte qu'un point en-ligne puisse être projeté sur sa position correspondante dans l'image scannée, et inversement chaque élément du tracé hors-ligne peut être temporellement indexé. Elle a été collectée auprès d'environ 700 scripteurs différents [**108**].

Dans nos travaux, nous nous intéressons aux bases de chiffres, lettres minuscules et majuscules isolés dont la répartition est donnée dans le tableau suivant.

Tableau 9 - Taille des sous-bases de caractères IRONOFF

| Base | Nombre de classes | Nombre total d'exemples |
|---|---|---|
| **Chiffres** | 10 | 4108 |
| **Minuscules** | 26 | 11868 |
| **Majuscules** | 26 | 11879 |

***La base IROCIEL***

La base IRONOFF est utilisée comme un complément à la base CIEL pour augmenter la variabilité des scripteurs dans le mode de génération de scripteurs virtuels. Nous nous référerons à l'unification des deux bases par le nom : IROCIEL.

### 3.4.2. Expressions synthétiques

Nous allons dans cette section détailler la constitution des bases d'expressions générées.

#### 3.4.2.1. La base calculette

Cette base est générée à partir du corpus semi-dynamique de cinq expressions générées avec le modèle « Calculette » pour chaque scripteur de la base isolée CIEL. Un total de 1 400 expressions sont générées dont 900 expressions sont utilisées pour l'apprentissage global du système, les 500 expressions restantes sont utilisées pour le test. Toutes les expressions sont en mode mono-scripteur. Le Tableau 10 montre la constitution de la base d'apprentissage et de test pour les symboles isolés et les expressions de la base. Ainsi, quelques expressions sont montrées dans la Figure 46.

Tableau 10 - La constitution de la base Calculette

| | | Base d'expressions | | Base de symboles |
|---|---|---|---|---|
| **Base** | **#Scripteurs** | **#Expressions** | **#Symboles** | **#Symboles** |
| **Apprentissage** | 180 | 180 × 5 = 900 | 5448 | 180 × 15 = 2700 |
| **Test** | 100 | 100 × 5 = 500 | 3051 | 100 × 15 = 1500 |

Figure 46 - Exemples de la base Calculette

### 3.4.2.2. La base RamanReduced

Nous avons généré deux bases à partir de la liste statique du corpus RamanReduced. La première, « ***RamanReduced_CIEL*** », est en mode mono-scripteur. Chacune des 36 expressions du corpus a été générée 280 fois en utilisant les jeux de symboles des 280 scripteurs de la base isolée CIEL. Le Tableau 11 montre la constitution de la base d'apprentissage et de la base de test des expressions.

La deuxième base « ***RamanReduced_IROCIEL*** » est générée afin de renforcer l'apprentissage global en introduisant plus de richesse dans les expressions. Toutes les expressions sont en mode scripteur virtuel. Dans ce cas, chacune de 36 expressions a été générée 200 fois en utilisant les jeux de symboles des scripteurs virtuels de la base IROCIEL Isolée. Seulement des échantillons supplémentaires des chiffres et lettres du corpus (15 classes) sont utilisés de la base IRONOFF. Nous obtenons donc près de 480 échantillons de chiffres et de lettres, en plus des 180 échantillons des autres classes qui sont moins fréquents dans le corpus. Quelques exemples d'expressions des bases synthétiques et de la base réelle sont montrés dans la Figure 43.

Tableau 11 - La constitution de la base RamanReduced_CIEL

| | | Base d'expressions | | Base de symboles |
|---|---|---|---|---|
| **Base** | **#Scripteurs** | **#Expressions** | **#Symboles** | **#Symboles** |
| **RamanReduced_CIEL Apprentissage** | 180 | 180 × 36 = 6480 | 180 × 412 = 74160 | 180 × 34 = 6120 |
| **RamanReduced_CIEL Test** | 100 | 100 × 36 = 3600 | 100 × 412 = 41200 | 100 × 34 = 3400 |
| **RamanReduced_IROCIEL Apprentissage** | 200 | 200 × 36 = 7200 | 200 × 412 = 82400 | 180 × 34 + 480 x 15 = 13320 |

### 3.4.3. Expressions réelles

L'objectif des bases réelles est de tester le système dans des conditions de production où chaque expression est écrite naturellement par un scripteur. A court terme, ces bases nous ont servi en tant que bases de test afin d'évaluer la performance du reconnaisseur d'expressions. Le deuxième but est de pouvoir aussi utiliser une bonne partie de la grande base « Wiki_CIEL » pour la phase d'apprentissage du système.

#### 3.4.3.1. La base RamanReduced Réelle

Nous avons dans un premier temps récolté le corpus Raman à l'échelle de notre équipe de recherche. Chacune des 62 expressions du corpus a été écrite deux fois, dix scripteurs ayant participés à cette base. Six scripteurs ont écrit 12 expressions, et les quatre autres ont écrit 13 expressions. Donc, nous avons obtenu un total de 124 vraies expressions dont 72 correspondent au corpus RamanReduced. Les scripteurs ont recopié les expressions imprimées sur un cahier de type Anoto. Un exemple du formulaire de collecte avec l'encre correspondant récoltée se trouve dans l'annexe B-5. Néanmoins, quatre expressions étaient inutilisables, soit à cause de défaut de stylo (problème de batterie ou d'appui), ou bien parce que l'expression écrite ne correspond pas à l'énoncé de l'expression. En conséquence, il nous reste 120 expressions au total dont 70 correspondant au corpus RamanReduced. Cet ensemble de 70 correspondante expressions au corpus RamanReduced sera désignée sous le nom de ***RamanReduced_réelle***.

#### 3.4.3.2. La base Wiki_CIEL

Dans un second temps, une collecte à plus grande échelle a été réalisée pour pouvoir obtenir une grosse base d'expressions qui puisse éventuellement servir à l'apprentissage et aussi bien sûr au test du système. Pour cela, nous avons filtré le corpus Wiki_CIEL en utilisant deux critères :

- Tous les symboles d'une expression doivent appartenir aux 223 symboles de la base isolée.
- La taille de l'expression (nombre de symboles) est comprise entre une longueur minimum et une longueur maximum, fixée ici respectivement à 3 et 49 symboles.

Il en résulte environ 57000 expressions mathématiques sur les 77000 expressions disponibles.

De ces expressions, 6144 expressions ont été choisies aléatoirement suivant quatre catégories. Trois catégories sont définies par la taille des expressions : courtes (3 à 8 symboles), moyennes (9 à 18 symboles) ou longues (19 à 49 symboles). Une quatrième catégorie « SuperCalc » est ajoutée correspondant aux expressions de type calculatrice scientifique (un vocabulaire de 36 symboles différents). Les formulaires de collecte sont imprimés sur des papiers de type Anoto, où chaque formulaire comporte six expressions, dont

une superCalc, deux courtes, deux moyennes et une longue. Des étudiants de l'université de Nantes, ainsi que des membres de notre équipe et quelques autres chercheurs ont participé à la saisie manuscrite de ces expressions pour un total de 512 scripteurs (2 formulaires par scripteur, soit 12 expressions par personne). L'annexe B-6 montre un exemple de formulaire de saisie. Chaque scripteur a également fourni un identifiant unique afin de pouvoir utiliser la base dans d'autres applications (ex : identification du scripteur), ainsi que son âge, sexe et la main d'écriture (droitier ou gaucher).

Ensuite, l'encre numérique a été récupérée et segmentée en expressions (l'encre récupérée d'un stylo représente toutes les pages saisies par ce stylo). Nous avons obtenu à la fin de la campagne les expressions séparées en fichier Unipen étiquetées au niveau expression par la chaîne LaTeX correspondante. L'impression des formulaires et la récupération des expressions ont été effectuées par notre partenaire industriel (Vision Objects).

Après une étape de contrôle de validité des données acquises, il est resté 5820 expressions valables (exclusion de fichiers non-valides ou mal saisis, néanmoins, il subsiste des expressions qui ont été saisies avec des variantes par rapport à ce qui était demandé). Nous allons ensuite pouvoir choisir des sous-ensembles de ces expressions en les filtrant à deux niveaux. Un sous ensemble doit être compatible avec le système que l'on cherche à évaluer. Un premier niveau de filtrage concerne le vocabulaire, on retient les expressions qui ne contiennent que les symboles acceptés par le classifieur du système. Le Tableau 12 montre la constitution de la base récoltée et des sous-ensembles correspondant aux corpus présentés précédemment. Les corpus ***RamanExtended*** et ***GarainExtended*** sont obtenus en ajoutant tous les chiffres ('0 à '9') et lettres ('a' à 'z', 'A' à 'Z') au vocabulaire de ces corpus dans l'esprit d'obtenir des corpus qui couvrent un nombre important de symboles. Les colonnes sont à interpréter comme suit : en ce qui concerne la base Raman_CIEL (par exemple) : 589 expressions filtrées par 53 symboles ont été imprimées, 543 ont été correctement récupérées, 540 correspondent à la vérité terrain.

Tableau 12 - La constitution de la base Wiki_CIEL

| Corpus | #classes de Symboles | #Expressions du corpus | #Expressions manuscrites récupérées | #Expressions correctemment saisies |
|---|---|---|---|---|
| Wiki_CIEL | 204 | 6144 | 5820 | -- |
| SuperCalc_Wiki_CIEL | 36 | 279 | 263 | 263 |
| RamanReduced_Wiki_CIEL | 35 | 276 | 252 | 247 |
| Raman_CIEL | 53 | 589 | 543 | 540 |
| RamanExtended_Wiki_CIEL | 69 | 1153 | 1065 | 1061 |
| GarainExtended_Wiki_CIEL | 113 | 1782 | 1684 | 1683 |

Un deuxième filtrage est indispensable pour s'assurer que la syntaxe des expressions corresponde à la grammaire retenue dans le système. En ce qui nous concerne, nous nous intéressons actuellement au sous ensemble ***RamanReduced_Wiki_CIEL***. Après le filtrage au niveau grammaire, nous avons 211 expressions qui vont être utilisées en tant que base de test.

Le filtrage au niveau symbole se fait facilement grâce aux scripts Perl[4]. Par contre, il est plus compliqué de filtrer les expressions qui correspondent à la grammaire, ceci a été effectué manuellement.

## 3.5. L'étiquetage de données manuscrites

Une question importante se pose lors de la construction de la base d'expressions : que doit-on définir dans la vérité terrain d'une expression mathématique manuscrite en-ligne ? Faut-il décrire la disposition des symboles, ou bien l'interprétation de l'expression ? Généralement le résultat de reconnaissance est en format standard de représentation d'expression mathématique, il s'agit souvent d'une chaîne LaTeX ou d'une structure MathML[5].

Le format de chaîne LaTeX est une représentation populaire d'une expression mathématique, mais il a ses limites. Tout d'abord, il ne représente que la disposition spatiale des symboles et il n'est pas destiné à interpréter l'expression. Deuxièmement cette représentation n'est pas unique : la même expression peut se décrire par plusieurs variantes de chaînes, notamment en ce qui concerne le nombre d'accolades par exemple. Considérons l'expression $(x+2)^3$, la chaîne LaTeX s'écrit soit comme : $(x+2)^3$ ou ${(x+2)}^{3}$ ou ….

De son coté, MathML est un format émergeant du langage XML, destiné à réaliser des expressions mathématiques dans les documents portables comme les pages web. De plus, il vise à encoder soit la disposition spatiale des symboles de l'expression ou l'interprétation (le contenu) de ces symboles. Dans le premier cas, la description représente une disposition unique des symboles, bien que dans le deuxième cas, la même expression s'affiche différemment selon le rendu utilisé. La Figure 47 montre les deux types de présentation MathML de l'expression $(x+2)^3$. Il est à remarquer que dans la deuxième description, les parenthèses ne sont pas explicites, contrairement à la description de disposition. En conséquence, la même description de contenu est équivalente à plusieurs représentations de disposition de la même expression.

[4] http://en.wikipedia.org/wiki/Perl
[5] http://www.w3.org/TR/MathML/

```
<msup>
    <mrow>
        <mo>(</mo>
        <mrow>
            <mi>x</mi>
            <mo>+</mo>
            <mn>2</mn>
        </mrow>
        <mo>)</mo>
    </mrow>
    <mn>3</mn>
</msup>
```

(a) Disposition

```
<apply>
    <power/>
    <apply>
        <plus/>
        <ci>x</ci>
        <cn>2</cn>
    </apply>
    <cn>3</cn>
</apply>
```

(b) Contenu

Figure 47 - Les deux formats de MathML pour l'expression $(x+2)^3$

Par exemple les trois expressions affichées dans la Figure 48 ont la même description de contenu défini par :

```
<apply>
    <minus/>
    <apply>
      <divide/>
      <ci>a</ci>
      <ci>b</ci>
    </apply>
</apply>
```

$$-\frac{a}{b} \qquad -(a/b) \qquad -\left(\frac{a}{b}\right)$$

Figure 48 - La même description MathML d'une expression rendu par trois applications différentes (a) Test Suite of MathML [6] (b) FireFox 3.5.7 and (c) MathMagic 4.81 [7]

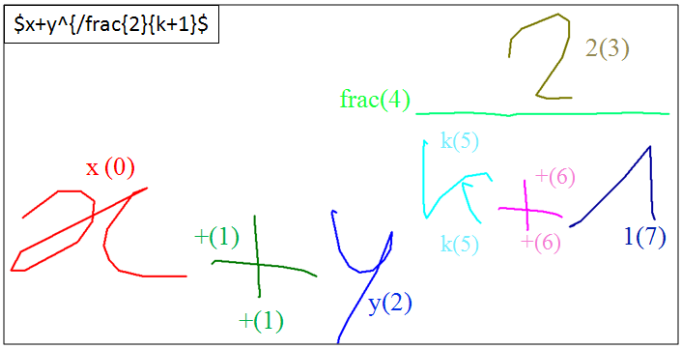
On constate de ces observations que la vérité terrain d'une expression mathématique ne doit pas être considérée seulement au niveau de l'interprétation mais aussi au niveau de la description de la disposition spatiale des symboles.

Puisque la sortie de notre système est au format LaTeX, nous avons choisi cette description comme vérité terrain principale des expressions. Donc, une chaîne LaTeX est associée à chaque expression correspondante, générée ou récoltée. Comme nous l'allons voir dans la section §3.6, l'étiquetage au niveau expression n'est pas suffisant pour bien évaluer la performance du système. En conséquence, chaque expression est étiquetée au niveau des traits comme le

[6] http://www.w3.org/Math/testsuite/mml2-testsuite/index.html
[7] http://www.mathmagic.com/

montre la Figure 49. Les traits du même symbole ont le même indice pour faire la différence entre différentes occurrences de la même classe, comme les symboles '+' dans l'exemple de la Figure 49.

Les expressions générées sont étiquetées automatiquement lors de la génération de la base d'expressions. Pour cela, au moment de l'ajout d'un symbole nous ajoutons en plus des coordonnées du symbole son étiquette et l'indice du symbole courant. Les étiquettes sont introduites dans le fichier Unipen par le mot clé .LABEL et l'indice est noté dans la ligne suivante. En ce qui concerne la base d'expressions récoltées, il s'agit d'une tâche longue et fastidieuse. Pour faciliter le processus, nous avons développé une méthode semi automatique pour accélérer l'étiquetage. Elle se base sur un classifieur de symboles appris sur la base de symboles isolés, et une pré-segmentation de l'expression en composantes connexes. Nous retenons ensuite l'étiquette donnée par le classifieur à chaque composante à condition qu'elle soit comprise dans le vocabulaire de l'expression courante. Ce processus est appliqué sur toutes les expressions de la base. Puis, nous vérifions manuellement les segmentations et les étiquettes, en faisant les corrections nécessaires. Cette méthode s'est montrée efficace et nous a permis d'étiqueter une bonne partie de la base récoltée.




```
.LATEX    $x+y^{\frac{2}{ k+1}}$
.PEN_DOWN
39.868331       225.014490      50.000000       62903.000000
52.737879       215.531665      92.000000       62943.000000
.....
98.797313       246.012173      36.000000       63249.000000
.PEN_UP
.LABEL    x
0
.PEN_DOWN
123.326222      224.277996      64.000000       28949.000000
.....
125.299385      253.382149      96.000000       29082.000000
124.806094      253.382149      36.000000       29095.000000
.PEN_UP
.LABEL    +
1
.PEN_DOWN
108.034210      241.543172      38.000000       29095.000000
109.020791      240.556590      84.000000       29108.000000
......
145.524304      245.982788      32.000000       29215.000000
.PEN_UP
.LABEL    +
1
.....
.....
.PEN_DOWN
238.406450      227.793148      0.000000        64128.000000
236.676344      228.225674      56.000000       64141.000000
.....
267.385729      225.630515      90.000000       64328.000000
.PEN_UP
.LABEL    1
7
```

Figure 49 - Expression étiquettée et son fichier Unipen

## 3.6. Comment évaluer un système de reconnaissance d'expressions mathématiques

L'objectif de l'évaluation d'un système de reconnaissance est de mesurer le degré de son succès. Elle est aussi indispensable pour pouvoir analyser et comprendre le comportement du système. Les mesures utilisées dans les deux cas sont de plusieurs natures, de très globales qui reflètent la performance totale du système, à très locales qui reflètent la performance des sous parties de systèmes.

***Taux de reconnaissance d'expressions (expRate)***

Pour évaluer un système de reconnaissance d'expressions mathématiques, il faut d'abord connaître la sortie prévue du système. Idéalement, la sortie doit être identique à la vérité terrain. La façon la plus simple est de compter les expressions bien reconnues en comparant la sortie avec la vérité terrain au niveau expression. Néanmoins, différentes vérités peuvent représenter la même expression. Donc, cette méthode d'évaluation n'est pas facilement automatisable. En conséquence, cette mesure est souvent calculée manuellement, c'est le cas dans plusieurs travaux [**15**][**85**][**94**]. Dans les cas où la représentation MathML est utilisée pour la sortie du système et aussi pour la vérité terrain des expressions [**99**], une évaluation automatique est possible. Le taux de reconnaissance d'expressions est calculé par la formule :

$$\text{expRate} = \frac{\text{nombre d'expressions bien reconnues}}{\text{nombre total d'expressions}} \qquad (1)$$

En ce qui nous concerne, en partant des chaînes LaTeX pour exprimer le résultat et la vérité terrain, nous avons proposé deux méthodes pour comparer ces chaînes. La première est semi-automatique car elle nécessite une intervention humaine ponctuelle. Pour limiter cette interaction nous sauvegardons dynamiquement une liste 1d'équivalence de chaînes LaTeX qui est mise à jour à chaque intervention auprès du vérificateur humain. Le format du fichier est simple, sur une première ligne se situe la vérité, et la chaîne à contrôler est sur la ligne suivante. Ensuite, sur la troisième ligne se trouve la valeur 0 si les deux chaînes ne sont pas équivalentes, et 1 dans le cas contraire, comme par exemple :

```
$/frac{x+y}{a+b^2}$
$/frac{x+y}{a+b_2}$
0
$/frac{x+y}{a+b^2}$
$/frac{x+y}{a+b^{2}}$
1
```

Afin de minimiser l'intervention humaine autant que possible, des conditions de rejet (nombre de symboles, symboles identiques) sont testées d'abord. Puis, si les deux chaînes partagent les mêmes symboles un couple

d'équivalence est cherché dans la liste des couples sauvegardés précédemment. Finalement, si la réponse n'y est pas trouvée, nous procédons à l'affichage des deux chaînes pour qu'un utilisateur décide si elles sont équivalentes ou pas. La réponse de l'utilisateur est ajoutée dans la liste qui est mise à jour. Les étapes de comparaison des deux chaînes sont montrées sur la Figure 50.

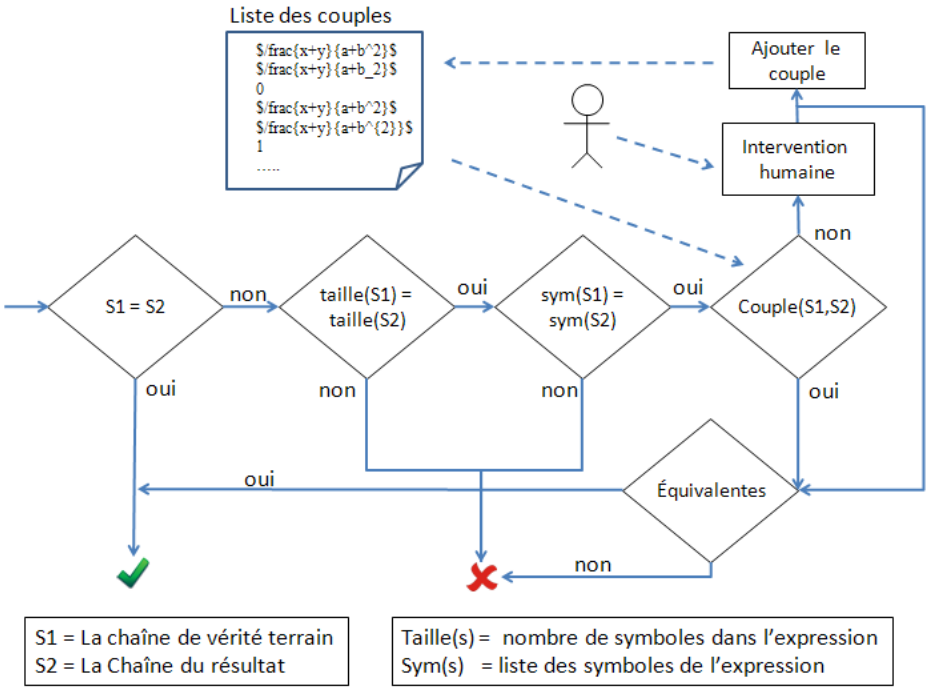



Figure 50 - Comparer deux chaînes LaTeX

Nous avons constaté dans nos expérimentations que le nombre de questions posées diminue au fur et à mesure des tests effectués. Plus on fait de tests, plus on ajoute de nouveaux couples dans le fichier. Pourtant, il y a toujours la possibilité qu'une intervention humaine soit demandée surtout lors du teste de nouvelles méthodes de reconnaissance qui peuvent produire de nouveaux formats de résultat non encore rencontrés.

Pour éviter cet inconvénient, nous avons proposé une méthode automatique consistant en transformer les chaînes LaTeX en format MathML (disposition) servant de format pivot pour la comparaison. Ainsi deux chaînes LaTeX différentes, mais donnant la même représentation MathML seront considérées équivalentes. La transformation se fait en utilisant un outil libre disponible sur Internet [8]. On s'assure que la représentation est unique en MathML en effectuant une normalisation du graphe intermédiaire représentant l'expression lors de sa transformation.

[8] http://gva.noekeon.org/blahtexml.

Ce taux de reconnaissance au niveau expression, quelque soit la façon de le calculer, est très global. Son inconvénient est qu'un seul symbole manqué, mal reconnu ou mal segmenté cause une reconnaissance erronée au niveau expression. Cela est surtout sensible quand les expressions à reconnaître sont longues, car alors il y a plus de chance de se tromper sur l'un des symboles. Cela n'empêche pas que beaucoup de travaux [**15**][**19**][**20**][**67**][**68**][**78**][**80**] utilisent cette mesure. Encore une fois, la comparaison est alors dangereuse. Ce n'est pas la même chose d'obtenir le même taux sur deux corpus dont les longueurs moyennes d'expressions sont différentes. Il est évident que les mesures calculées au niveau expression sont très globales et peuvent cacher des problèmes complexes. De telles mesures ne permettent pas de comprendre la performance du système dans les différents niveaux intermédiaires. Un exemple similaire est celui de la reconnaissance de phrases. En effet, quand une phrase est très longue, il est plus probable qu'elle contienne des erreurs de mots. Il est donc commun dans ce cas de fournir plutôt un taux de reconnaissance au niveau mot. De plus, en fonction de la structure du système, il peut être intéressant d'examiner un comportement spécifique. Par exemple, un système avec une analyse syntaxique et structurelle mérite d'être évalué aux niveaux segmentation et reconnaissance.

Par conséquent, des mesures plus en profondeur sont souhaitables. Dans ce cadre, il existe deux familles principales de mesures : qualitatives et quantitatives qui sont très souvent calculées automatiquement.

### 3.6.1. Mesures quantitatives

Plusieurs indicateurs complémentaires sont proposés pour évaluer la performance d'un reconnaisseur d'expressions mathématiques. Un des indicateurs les plus utilisés [**15**][**20**][**23**][**24**][**78**][**79**][**80**][**97**][**99**][**109**] et des plus directs est le ***taux de reconnaissance des symboles (recoRate)*** qui est le taux des symboles bien reconnus dans l'expression :

$$\text{recoRate} = \frac{\text{nombre de symboles bien reconnus}}{\text{nombre total de symboles}} \qquad (2)$$

Par exemple, si l'expression $x^2+i$ est reconnue *x2+1* , le taux obtenu est alors de *3/4 = 0.75, car* il y a une seule erreur de reconnaissance, le *'i'* ayant été reconnu comme un *'1'*. Il est évident que cet indicateur est facile à calculer, mais en quelque sorte il ne reflète que la performance du classifieur de symboles. Par exemple, si l'expression précédente était reconnue comme $\alpha^3 x1$, alors ce taux indiquerait 0% de taux de reconnaissance des symboles et aussi 0% au niveau d'expression. Mais cela ne veut pas dire que le système a totalement échoué. Pour surmonter cette limitation, une autre mesure est largement utilisée, c'est ***le taux de segmentation*** [**15**][**20**][**23**][**24**]**.** Par définition, ce taux est le pourcentage de symboles bien segmentés :

$$\text{segRate} = \frac{\text{nombre de symboles bien segmentés}}{\text{nombre total de symboles}} \quad (3)$$

Un symbole est dit bien segmenté si ses traits (ou pixels) sont regroupés ensemble sans s'occuper de savoir si ils sont bien reconnus ou non. Revenant à l'exemple ci-dessus, si on considère le résultat de segmentation montré dans la Figure 51-a, on aura un taux de segmentation de 100%, signifiant que tous les segments des symboles sont bien extraits. Si la reconnaissance de l'expression échoue alors, c'est que la faiblesse du système provient soit du classifieur, soit du système d'interprétation.

En effet, comme c'est le cas dans la reconnaissance de textes, les taux de reconnaissance et de segmentation de symboles sont interdépendants. Par exemple, supposons que l'expression $x^2+i$ soit reconnue comme $x^2+1$ à cause du mauvais regroupement du point du *'i'* dans la dernière composante dans la Figure 51-b. Dans ce cas, le taux de segmentation est de 50% (2/4) et le taux de reconnaissance est aussi affecté avec un taux de 75% (3/4). On a toujours le taux au niveau expression qui reste à 0%. Cette dépendance de ces trois taux montre la nécessité de s'intéresser conjointement à ces différentes grandeurs.

Une autre propriété intéressante du résultat de reconnaissance, est de savoir à quel point la structure de l'expression est bien reconnue. Supposons que l'encre dans la Figure 51-a soit bien segmentée (100%), mais reconnue comme $c^3x1$. Le classifieur de symboles est alors très faible (0% de symboles bien reconnus). Mais le système a réussi à trouver la bonne structure de l'expression, ce qui indique que la phase d'interprétation s'est bien déroulée. Pour cette raison, il est intéressant d'avoir accès à ce type d'information afin de mieux comprendre le comportement du système.

a) Segmentation correcte b) Segmentation erronée

Figure 51 - Résultat de la phase de segmentation de l'expression $x^2+i$

Contrairement aux taux de segmentation et de reconnaissance de symboles, la mesure structurelle dépend de l'architecture du système et de sa représentation de l'expression mathématique. Dans ce cadre, on peut vérifier que les relations entre les symboles sont correctement interprétées [**23**] [**78**][**80**], ou que certaines structures, telles que la somme, la racine carrée sont bien reconnues [**99**][**109**]. Dans [**24**][**85**] cette mesure est calculée en fonction des lignes de base correctement trouvées dans l'expression reconnue. Ces mesures d'évaluation structurelle ignorent la performance du classifieur de symboles et se focalisent sur l'exactitude de l'analyse structurelle.

La mesure structurelle est moins présente dans la littérature pour plusieurs raisons. Tout d'abord, les relations spatiales ne sont pas toujours explicites dans les structures reconnues. De plus, quand les relations spatiales sont présentes dans l'arbre résultat, la mise en correspondance des relations requiert des algorithmes complexes de comparaison d'arbres. Ceci est possible en calculant une distance d'édition entre deux arbres [**110**]. Mais cette distance ne considère que l'insertion, la suppression, et le remplacement et pas l'inversion à cause de la complexité de l'opération de mise en correspondance. En effet, le problème du matching d'arbres non-ordonné est un problème NP complet [**18**]. En conséquence, dans un cas d'inversion du numérateur et du dénominateur au niveau de l'interprétation d'une fraction, comme dans l'exemple $(a+b)/(c+d)$ reconnue comme $(c+d)/(a+b)$, il y aura une distance très élevée même s'il n'y a qu'une seule erreur d'inversion.

Nous proposons deux concepts au niveau des relations spatiales : relation correcte et relation trouvée. Une ***relation correcte*** est une relation trouvée dans l'arbre de résultat avec une correspondance dans la vérité terrain impliquant les mêmes symboles dans les deux relations. Si les symboles sont différents, la relation est considérée comme une ***relation trouvée***. De plus les relations restantes qui n'ont pas de correspondances dans la vérité terrain sont des ***relations non-valides***. Les taux de reconnaissance de relations spatiales correctes et trouvées sont définis par les formules suivantes :

$$\text{relRateCorrect} = \frac{\text{nombre de relations correctes}}{\text{nombre total de relations}} \quad (4)$$

$$\text{relRateFound} = \frac{\text{nombre de relations trouvées}}{\text{nombre total de relations}} \quad (5)$$

Ces deux mesures ne prennent en compte que les relations spatiales sans considérer la segmentation ni la reconnaissance des symboles. En effet, nous pouvons considérer trois degrés de liberté. La ***mesure libre*** de taux relationnel est indépendante de la segmentation et la reconnaissance. Alors, elle ne reflète que la capacité de la phase d'analyse structurelle du système. La ***mesure de 2$^{ième}$ degré*** considère en plus de la relation, la segmentation correcte des éléments de la relation. Plus strictement, La ***mesure de 3$^{ième}$ degré*** requiert que les symboles des segmentations trouvées soient aussi bien reconnus. Cette dernière mesure reflète la qualité de la reconnaissance d'expressions car elle prend en compte toutes les phases de reconnaissance d'une façon intégrée. Revenons à l'exemple de la Figure 51, si l'expression est reconnue comme $c^3 \times 1$ en considérant la segmentation (a), avec les trois degrés de liberté on obtient respectivement : 100% (les deux relations : indice et opérateur sont trouvées), 100% (les segments des relations sont correctes), et 0% (aucune relation n'a les bons symboles). Tandis qu'en considérant la segmentation du cas (b), il n'y a que la relation indice correctement reconnues, ce qui donne donc 100%, 50%, 0% comme taux de reconnaissance de relations aux trois

degrés. Malheureusement, nous n'avons pas pu mettre en œuvre automatiquement ces dernières mesures car leur calcul nécessite de mettre au point des algorithmes complexes de comparaison d'arbres.

Chacune des mesures introduites ci-dessus reflète la performance de tout ou partie du système de reconnaissance d'expressions mathématiques. Dans la section suivante, nous allons discuter de quelques mesures intégrées pour évaluer globalement un système au moyen d'un indice de performance. Ce type de mesures se focalise sur la qualité de la reconnaissance sans chercher à incriminer chacune des sous parties du système (segmentation, reconnaissance, interprétation).

### 3.6.2. Mesures qualitatives

Une simple mesure intégrée a été utilisée par [**78**] dans le but de mesurer la totalité du système avec une seule valeur $R_s$. Cette mesure prend en compte le nombre de symboles et d'opérateurs (implicites et explicites) bien reconnus et est calculée par la formule :

$$R_s = \frac{\text{nombre de symboles et opérateurs bien reconnus}}{\text{nombre total de symboles et opérateurs}} \ ;$$

Cette mesure intègre la performance du classifieur et de l'interprétation. Une mesure très similaire est proposée par [**103**]. Mais au lieu de compter les structures bien reconnues, on demande l'avis de personnes qualifiées (Étudiants d'université). Ces personnes évaluent la structure de l'expression reconnue avec une valeur entre 0 et 1 (1 pour totalement correcte). La somme de ces valeurs est ensuite utilisée au lieu du nombre d'opérateurs bien reconnus. Il est clair qu'une telle mesure ne semble pas très efficace. Elle requiert une intervention humaine à chaque expression reconnue. De plus, il est évident que le résultat est subjectif et dépend fortement des personnes participant à l'évaluation. Il est donc souhaitable de disposer de mesures plus automatisées.

Une mesure intégrée de performance de reconnaisseur d'expressions mathématiques a été proposée dans [**86**][**101**]. La segmentation, la reconnaissance et les structures sont comparées en utilisant la présentation MathML de l'expression de sortie et de référence. Ensuite, un indice ***de performance γ*** dans l'intervalle [0..1] est calculé automatiquement pour estimer l'exactitude de l'expression reconnue (1 pour une expression bien reconnue). Il considère à la fois le taux de reconnaissance de symboles et aussi le positionnement de ces symboles sur les lignes de base de l'expression. La ligne principale est le niveau 0, et une erreur y est considérée beaucoup plus grave qu'à un niveau supérieur (exposant) ou inférieur (indice).

L'indice de performance est défini par :

$$\gamma = 1 - \frac{S_e + \sum_i O_i \times \frac{1}{|i|+1}}{S_t + \sum_i R_i \times \frac{1}{|i|+1}} ;$$

Pour une expression donnée, $S_t$ est le nombre total de symboles et $S_e$ est le nombre de symboles mal reconnus, $R_i$ le nombre de symboles du niveau *i* et $O_i$ le nombre de ceux mal positionnés du niveau *i*. Puis la performance du système est la moyenne de l'indice $\gamma$ sur toute la base du test. Une autre mesure a été proposée dans [**111**], les arbres de la représentation MathML de la référence et du résultat sont transformés en chaîne d'Euler. Puis on calcule la distance d'édition entre les deux chaînes pour estimer la mesure d'évaluation.

Le grand avantage d'une mesure intégrée est de pouvoir effectuer son calcul d'une façon automatique. La complexité des deux dernières méthodes est raisonnable, et donc applicable aux grandes bases d'expressions. De plus, elles sont indépendantes de l'architecture du système et permettent donc de comparer des systèmes différents. Par contre, ces mesures intégrées ne sont destinées qu'à mesurer la globalité du système. Elles ne sont pas adaptées quand on cherche à mesurer la performance d'une sous partie du système. Car même si l'indice indique une faiblesse, il ne porte aucune information sur son origine. De telles indicateurs sont donc complémentaires des mesures plus locales.

## 3.7. Conclusion

Nous avons dans ce chapitre étudié le problème d'évaluation d'un système de reconnaissance d'expressions mathématiques.

Dans un premier temps nous avons détaillé les différentes étapes mises en jeu afin de créer de manière synthétique une expression manuscrite en-ligne. A l'aide de l'arbre de dérivation de la chaîne LaTeX, il est possible de définir les différentes boîtes englobantes associées à chaque symbole en fonction des classes de nœuds rencontrées. Cette méthode nous fournit un outil intéressant qui permet de produire des grosses bases d'expressions de manière aisée. Le choix du corpus d'expressions et de la taille de la base est critique pour bien évaluer le système. Différentes bases et corpus sont proposés dans la littérature. Cette diversité ne permet pas de comparer les différents systèmes. De plus, les mesures utilisées pour l'évaluation ne sont pas standardisées, ce qui complique encore plus la comparaison des systèmes. Pour permettre aux autres chercheurs à se comparer avec nos résultats, nous allons rendre publique la base ***RamanReduced_Wiki_CIEL*** de 211 expressions.

Ensuite, nous avons décrit les caractéristiques à considérer pour construire notre base d'expressions. Nous avons défini trois corpus d'expressions : Calculette, RamanReduced, et Wiki_CIEL pour construire nos bases d'expressions pour l'apprentissage et le test de notre système. Finalement, nous avons exploré les mesures d'évaluations les plus souvent utilisées dans ce domaine. Plus précisément, nous avons retenu le taux de reconnaissance d'expressions, le taux de segmentation, et le taux de reconnaissance des symboles.

Nous allons dans le chapitre suivant détailler l'architecture de notre système de reconnaissance d'expressions mathématiques manuscrites en-ligne. Nous allons également présenter les algorithmes et les stratégies mises en œuvre pour la conception, l'apprentissage, et le test du système.

# Chapitre 4 : Système de reconnaissance d'expressions mathématiques manuscrites en-ligne

## 4.1. Architecture globale du reconnaisseur d'expressions mathématiques

Les chapitres précédents nous ont montré que l'architecture globale d'un système de reconnaissance d'expressions mathématiques pouvait se décomposer fonctionnellement en une séquence d'étapes fortement interdépendantes. L'objectif de l'architecture que nous allons proposer est de compenser les problèmes de chacune des étapes individuelles de la reconnaissance. Nous considérerons la reconnaissance d'expressions mathématiques comme une optimisation simultanée de la segmentation, de la reconnaissance de symboles, et de l'interprétation.

Considérons une expression donnée comme un ensemble de traits représentant des symboles :

$$E = \{t_1, t_2, ..., t_n\} \quad (6)$$

Reconnaître une expression consiste en particulier à trouver le meilleur regroupement possible de ses traits, à identifier le symbole correspondant à chaque regroupement, et finalement à interpréter l'expression dans le langage du domaine. Ces différentes étapes vont participer à la définition d'une fonction de coût $C_E$ que l'on va chercher globalement à minimiser. Ce coût est déduit du score de reconnaissance d'hypothèses de segmentation ($C_{reco}$) et des coûts structurels ( $C_{struct}$ ) entres les symboles de l'expression. Le reconnaisseur d'expressions est présenté dans la Figure 52, il est également utilisé dans le système d'apprentissage (avec ou sans modèle de langage) comme nous l'allons voir. Il consiste en :

- Un générateur d'hypothèses de symboles élaborant des combinaisons des traits. Un groupe de traits s'appelle une hypothèse de symbole *'hs'* où $hs \subseteq E$ .
- Un classifieur de symboles associant un score de reconnaissance, mais aussi une étiquette pour chacune des hypothèses. Il peut être un classifieur simple ou hybride, avec ou sans classe de rejet.
- Un analyseur de structure permettant d'obtenir des informations structurelles sur chaque hypothèse afin d'établir un coût structurel.
- Un modèle de langage défini par la grammaire de production d'expressions mathématiques.
- Un bloc de décision choisissant l'ensemble des hypothèses minimisant le coût global et respectant le modèle de langage :

$$R(E) = E'; C_{E'} = \arg\min(f(C_{reco}, C_{struct})) \quad (7)$$

Sachant que :

$$E' = \{hs_1, hs_2, ..., hs_n\}; hs_i \cap hs_j = \phi \text{ et } \bigcup_{i=1..n} hs_i = E \quad (8)$$

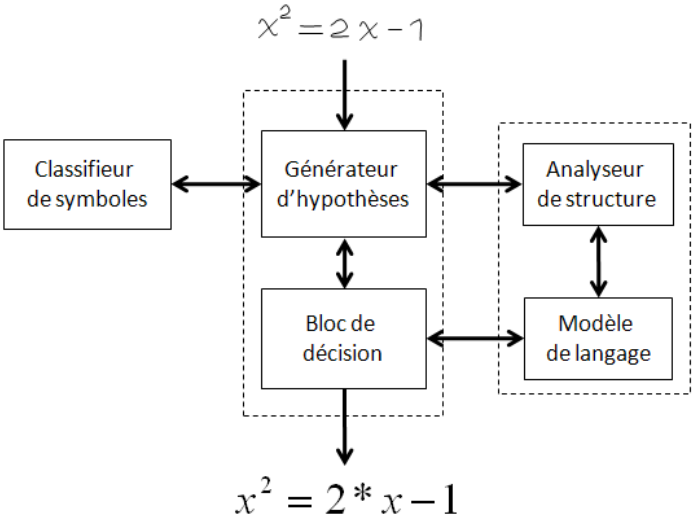



Figure 52 - Architecture du système

Dans cette architecture, les briques correspondant au générateur d'hypothèses et d'analyse syntaxique (modèle de langage) seront fournies par notre partenaire industriel « Vision Objects ». Nous nous intéresserons à mettre au point le classifieur de symboles, et l'analyseur structurel de sorte qu'ils soient bien adaptés au problème de reconnaissance d'expressions mathématiques. Nous proposerons également plusieurs stratégies d'apprentissage du système à partir de bases de symboles isolés et/ou d'expressions mathématiques complètes.

## 4.2. Générateur d'hypothèses de symboles

Le générateur d'hypothèses élabore des combinaisons des traits. Un groupe de traits s'appelle une hypothèse de symbole ( $hs$ ). Toutefois, beaucoup d'hypothèses de segmentation sont invalides. Celles-ci correspondent à deux cas : la sur-segmentation ou bien la sous-segmentation. Dans le premier cas, seul un sous-ensemble des traits du symbole, lorsque celui-ci est écrit en plusieurs traits, est sélectionné ; tandis que dans le second cas, le regroupement considéré fusionne des traits appartenant à plusieurs symboles. La Figure 53 montre des exemples de bonnes ou mauvaises segmentations. Des exemples d'hypothèses valides de segmentation sont entourés en vert. Des hypothèses invalides de segmentation sont entourées en rouge pointillé. Des symboles entiers ont pu être fusionnés, c'est le cas du '2' et du 'a' qui sont considérés comme un seul symbole. Il est possible aussi de fusionner des sous parties de plusieurs symboles, comme le cas du 'x' et de la barre supérieure de du caractère '='. Ces deux cas sont des sous-segmentations. Un exemple de

sur-segmentation qui ne considère qu'une partie d'un symbole, est montré avec la barre inférieure du caractère '='.

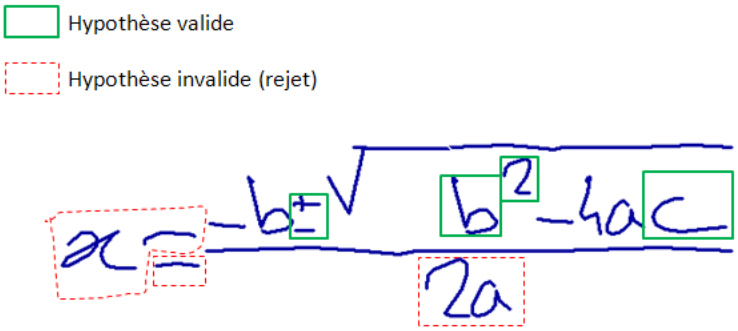



Figure 53 - Exemple des hypothèses de segmentation

Le générateur est basé sur une extension d'un algorithme de programmation dynamique en deux dimensions (2D-DP) pour générer les hypothèses de symboles. Le 2D-DP compense les limites d'un algorithme 1D-DP qui ne permet que de fusionner des traits temporellement consécutifs en permettant de gérer des regroupements de traits non-consécutifs. Ceci est très important dans la reconnaissance d'expressions mathématiques car il est fréquent de saisir quelques traits tardifs pour corriger une erreur, ou par habitude du scripteur (point du 'i' par exemple qui s'écrit à la fin). L'introduction de l'aspect 2D augmente exponentiellement le nombre d'hypothèses par rapport au nombre de traits. Pour contrôler et limiter le nombre d'hypothèses, les contraintes suivantes ont été définies afin de réduire autant que possible les hypothèses invalides :

a. Nombre maximal d'hypothèses de symboles $N_h$ : comme nous l'avons vu dans le chapitre 2, le nombre possible d'hypothèses est défini par le nombre de Bell. Le nombre d'hypothèses de segmentation d'une simple expression par exemple de 7 traits est de 877. Ce nombre augmentant très rapidement avec le nombre de traits. Nous introduisons un nombre maximal d'hypothèses pour borner l'explosion combinatoire. Nous avons fixé expérimentalement ce nombre à 1500 hypothèses. Ce nombre est variable selon le domaine concerné par le reconnaisseur, 500 hypothèses sont suffisantes pour un reconnaisseur d'expressions de type calculette (cf. chapitre 3).

b. Nombre maximal de traits par hypothèse $N_a$ : le nombre de traits par symbole varie selon les symboles mathématiques. Les petits symboles s'écrivent le plus souvent en un seul trait (ex : −, α). D'autres peuvent atteindre jusqu'à 7 traits (ex : arctan). En étudiant la distribution du nombre de traits par symboles sur la base des symboles isolés, nous avons fixé Na = 5, qui semble suffisant pour les classes des symboles dans les corpus choisis (moins de 0.4% des symboles ont plus de 5 traits).

c. Distance maximale entre les traits $D_{max}$ : tous les traits composant un symbole se situent près l'un de l'autre. Il est donc possible d'utiliser cette contrainte pour éliminer les regroupements de traits éloignés.

d. Nombre maximal de sauts temporels $N_j$ : un saut temporel a lieu quand un symbole est complété après avoir saisi d'autres symboles (par exemple pour revenir en arrière à la fin d'une expression pour compléter certains symboles multi-traits comme le 'i', le 't',…). En conséquence, il n'est pas rare d'avoir un saut temporel à l'intérieur d'une expression. Nous autoriserons donc un maximum de deux sauts temporels par symbole.

Les paramètres ci-dessus sont peu contraignants où 99,7% des vrais symboles sont proposés par le générateur d'hypothèses, comme nous allons voir dans l'expérimentation présentée dans la section 5.7.1.

## 4.3. Classifieur de symboles

Nous souhaitons que le classifieur de symboles associe non seulement une étiquette à chaque hypothèse de segmentation mais également un score de reconnaissance. Pour cela nous avons considéré des classifieurs avec sorties probabilistes tels qu'il est possible d'en obtenir à l'aide d'un étage de sortie approprié que cela soit avec un réseau de neurones ou un SVM. Pour une hypothèse donnée, $p(C_j|hs_i)$ dénote la probabilité que l'hypothèse $hs_i$ soit la classe $C_j$ ; avec $\sum_j p(C_j|hs_i) = 1$.

Nous avons vu que les symboles mathématiques comportent des ambiguïtés qui peuvent être résolues au niveau du contexte global de l'expression. Pour cette raison, quelques méthodes proposées considèrent les N meilleurs candidats du classifieur de symboles [**24**]. D'une façon similaire, d'autres approches retardent la décision sur l'identité des symboles ambiguës et cherchent à les fixer globalement [**15**]. Dans cette optique, nous allons déterminer le nombre de candidats retenus pour chaque hypothèse $hs_i$. Nous retenons *topN* candidats avec un maximum de *N* ($topN \leq N$). La valeur de *topN* est choisie par la condition :

$$\sum_{j=1}^{topN} p(C_j|hs_i) \leq k \qquad (9)$$

Le but du seuil *k* est de ne garder que les candidats avec une bonne confiance et ainsi d'éviter de garder les *N* candidats systématiquement. Le choix de la valeur du *'k'* est déterminé expérimentalement comme nous allons le voir dans la section 5.4.2.

Le score de reconnaissance est ensuite transformé en coût en utilisant une fonction logarithmique. Le coût que l'hypothèse courante soit de la classe *j* est alors défini par :

$$C_{reco}(hs_i) = -log(P(c=C_j|hs_i)) \quad (10);$$

où plus la probabilité est élevée, plus la fonction de coût est faible. Avec un tel type d'approche, une situation importante à prendre en compte est celle du comportement du classifieur vis-à-vis des hypothèses invalides. En particulier, la capacité de rejet du classifieur doit être examinée (rejet d'ignorance et non d'ambiguïté).

Dans une application classique, l'entrée du classifieur est une forme qui représente un exemple de son vocabulaire. Par exemple, dans un reconnaisseur de chiffres, l'entrée est toujours supposée être un signal manuscrit correspondant à l'un des dix chiffres. Néanmoins, quand une forme ne correspondant à aucun chiffre est présentée en entrée du classifieur, la sortie est toujours un chiffre. Dans notre application, cela est surtout gênant lorsque nous voulons faire la différence entre les vraies classes (les symboles bien segmentés) et les formes indésirables (les mauvaises segmentations).

Avant de détailler les structures des classifieurs utilisés, nous allons présenter les prétraitements appliqués à chaque hypothèse de symbole. Ils en résultent un vecteur de caractéristiques servant en tant qu'entrée du classifieur.

### 4.3.1. Prétraitements

En fait, un signal manuscrit comporte une grande variabilité selon la vitesse (nombre de points) et la taille de l'écriture. Entraîner un classifieur avec de telles données augmente notamment la difficulté de sa conception et de son apprentissage. Il est donc important de prétraiter l'entrée du classifieur pour faciliter sa tâche de reconnaissance. Cette étape de prétraitements vise donc à diminuer la variance intra-classe et inversement à augmenter la variance inter-classes afin de favoriser la séparation des classes.

#### 4.3.1.1. Ré-échantillonnage et normalisation

Une première grande variation entre des signaux en-ligne est le nombre de points d'échantillonnage. Cette variation vient de l'outil de saisie ou de la vitesse de l'écriture. Plus la vitesse est élevée moins il y a de points d'échantillonnage. En effet, selon les scripteurs, le temps de formation d'une lettre peut être plus ou moins long et la forme des symboles très différente. Cette étape consiste donc à réduire l'information redondante, lorsqu'il y a trop de points, ou à étendre l'information lorsqu'il y a peu de points. Il est ainsi possible d'avoir un nombre uniforme de points pour toutes les entrées du classifieur.

Pour cela à partir de la représentation brute du signal d'entrée, on analyse les coordonnées spatiales (x, y) et les informations de lever et poser de stylo

(.Pen_UP et .Pen_Down) qui caractérisent un nouveau trait. On ré-échantillonne le signal en un nombre *N* fixe de points. On distingue deux types de points : les points où le stylo est posé (vrais points), et les points où le stylo est levé pour se déplacer d'un trait à un autre (points imaginaires). Les nouveaux points sont caractérisés par la matrice de dimension 3xN [X,Y, PenUpDown] où X et Y sont les coordonnées des nouveaux points sur les axes des abscisses et des ordonnées. La troisième dimension spécifie l'état du stylo à un point donné (posé ou levé). Cette information est codée de la façon suivante :

$$PenUpDown = \begin{cases} -1 \text{ ; stylo levé} \\ 1 \text{ ; stylo posé} \end{cases}$$

Sur la Figure 54, qui illustre quelques exemples de symboles avant et après ré-échantillonnage spatial, les points caractérisant l'appui du stylo (PenUpDown=1) sont présentés par les points bleus, les autres par des carrés rouges (dans l'exemple du *b* à deux traits).

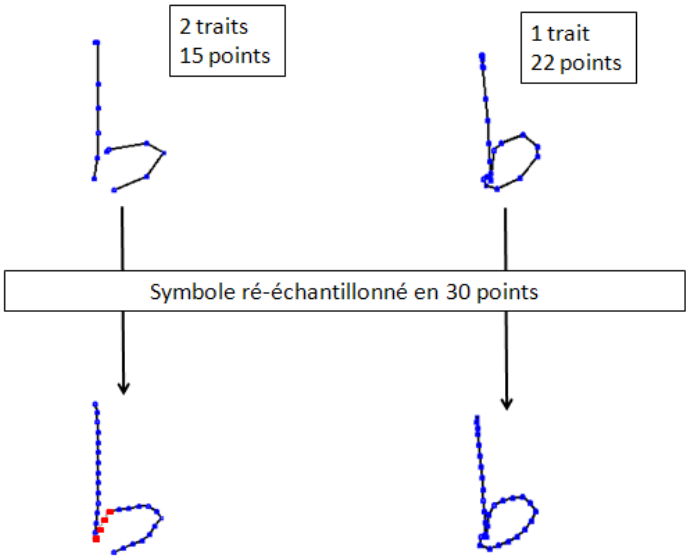



Figure 54 - Exemples du ré-échanthillonnage du symbole 'b'

Les symboles échantillonnés sont ainsi représentés comme une séquence de points [X(n),Y(n)] (n numéro du point) espacés par une distance régulière selon la longueur du tracé, par opposition au temps dans la séquence d'entrée. Par expérimentation, nous avons fixé le nombre de points à 50 pour chaque hypothèse de symbole et utilisé une simple interpolation linéaire.

Avant d'analyser l'information acquise pour extraire les caractéristiques du signal, il est nécessaire de recentrer le signal manuscrit et de le normaliser. Le but du recentrage et de la normalisation est d'obtenir une représentation invariante aux translations, aux distorsions spatiales, et à la taille des symboles. Le signal est d'abord normalisé et centré par rapport à sa taille

maximale dans un carré [−1, +1]×[−1, +1]. Ensuite dans le but de faciliter la tâche du classifieur, on extrait de cette séquence de points des informations géométriques locales telles que la direction du mouvement et la courbure de la trajectoire. On obtient alors une séquence de vecteurs de 7 caractéristiques par point, décrites dans la section suivante.

#### 4.3.1.2. Les caractéristiques utilisées pour la reconnaissance

Comme indiqué précédemment, nous avons fixé le nombre de points à 50 pour chaque hypothèse de symboles. En chacun de ces points, un vecteur de 7 caractéristiques est extrait. Il comporte les coordonnées x et y, les cosinus directeurs de la direction (cosθ, sinθ) et de la courbure (cosΦ, sinΦ) de la trajectoire et l'état posé ou levé du stylo en ce point, cf. la Figure 55.

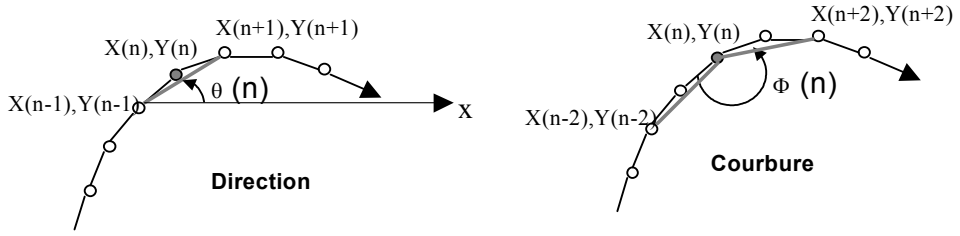



Figure 55 - Illustration des angles dans le calcul de la direction (angle θ) et de la courbure de la trajectoire (angle Φ) [**74**]

Le cosinus directeur de la direction (cosθ, sinθ) représente la direction de la trajectoire du tracé selon les axes X et Y respectivement où :

$$Direction_x(n) = \cos\theta(n) = \begin{cases} 0 & ; X(n-1) = X(n+1) \\ \dfrac{distance(X(n-1), X(n+1))}{distance(P(n-1), P(n+1))} & ; \text{autrement} \end{cases}$$

$$Direction_y(n) = \sin\theta(n) = \begin{cases} 0 & ; Y(n-1) = Y(n+1) \\ \dfrac{distance(Y(n-1), Y(n+1))}{distance(P(n-1), P(n+1))} & ; \text{autrement} \end{cases}$$

Le cosinus directeur de la courbure (cosΦ, sinΦ) représentent la courbure de la trajectoire du tracé selon les axes X et Y respectivement où :

$$Courbe_x(n) = \cos\Phi(n) = Direction_x(n-1) \times Direction_x(n+1) + Direction_y(n-1) \times Direction_y(n+1)$$

$$Courbe_y(n) = \sin\Phi(n) = Direction_y(n-1) \times Direction_x(n+1) - Direction_x(n-1) \times Direction_y(n+1)$$

La direction et la courbure de premier et dernier points sont identiques à leurs points voisins. Il est à remarquer que ces caractéristiques sont sensibles au sens de l'écriture et au nombre de traits. Le sens de l'écriture est surtout bénéfique pour différencier les symboles qui se ressemblent visuellement,

mais s'écrivent différemment, comme '$a$' et '$\alpha$' : a et alpha (cf. chapitre 1 pour plus d'exemples).

On obtient ainsi une matrice de taille fixe 50×7 représentant un symbole. Cette matrice est ensuite propagée dans le classifieur. A la sortie du classifieur, chaque sortie fournit la probabilité du label associé qui va servir pour déduire le score de reconnaissance de l'hypothèse courante du symbole.

### 4.3.2. Les réseaux de neurones

Nous adoptons les réseaux de neurones en tant que classifieur principal de notre système [**112**]. Un neurone formel est capable de réaliser la tâche de séparation de deux classes linéairement séparable. Comme le montre la Figure 56, un neurone j est défini par les paramètres suivants :

- un ***vecteur d'entrées*** $X_j = (x_i)_N$ de taille N fixée
- un ***vecteur de poids*** $W_j = (w_{ji})_N$
- ***un biais***, dit aussi seuil qui peut être traité comme un poids supplémentaire auquel on présente une activité constante égale à 1
- un ***potentiel synaptique*** $v_j$, avec $v_j = \sum_i w_{ji} \times x_i$
- une ***fonction d'activation*** appelée aussi fonction de transfert $f$. Une forme analytique très courante pour la décision est la fonction sigmoïde, mais d'autres fonctions peuvent également être utilisées (comme la fonction Softmax).
- une ***sortie*** $y_j = f(v_j)$. La sortie de cette cellule est ainsi une fonction non linéaire de la somme pondérée de ses entrées.

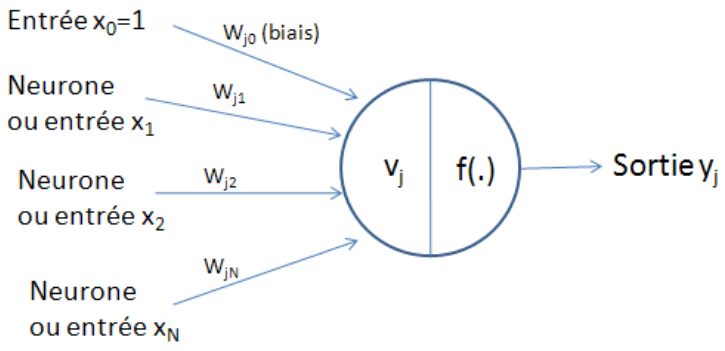



Figure 56 - Modélisation d'un neurone

La multiplication et la combinaison de neurones permet de séparer plusieurs classes linéairement séparables. Dans un problème de reconnaissance de l'écriture, les classes de symboles sont inséparables linéairement dans

l'espace de caractéristiques. Ce problème peut être résolu en utilisant des réseaux à plusieurs couches permettant de séparer les classes par d'autres frontières que des hyperplans dans l'espace de caractéristiques. Les vrais défis résident dans le choix de la topologie du réseau et dans le réglage des poids du réseau (apprentissage).

Les réseaux de neurones sont capables de reconnaître des formes globales, telles que les symboles mathématiques avec une bonne capacité de généralisation. Les réseaux de neurones ont la capacité de stocker de la connaissance et la rendre disponible à l'usage. La connaissance est stockée dans les poids du réseau, obtenus par un processus d'apprentissage.

#### 4.3.2.1. Le perceptron Multi couches (PMC)

Un réseau PMC comporte trois types de couche : la couche d'entrée, la ou les couches cachées et la couche de sortie. Dans ce type d'architecture, les neurones d'une couche sont complètement connectés à ceux de la couche suivante. Les couches cachées ont pour but d'extraire de l'espace d'entrée de l'information pertinente pour résoudre le problème de décision. Quant à la couche de sortie son rôle est de prendre la décision de reconnaissance, elle possède autant de neurones qu'il y a de classes de symboles (ex : 10 dans le cas du classifieur des chiffres 0 à 9). Pour obtenir des sorties de type probabilités (entre 0 et 1) [**113**], on utilise en sortie une fonction de transfert dite Softmax :

$$Soft\,max(v_i) = P(Classe = C_i \,|\, X = x) = \frac{e^{v_i}}{\sum_k e^{v_k}} \;;$$

avec :

$i$ : indice du neurone correspondant à la classe Ci

$k$ : indice balayant tous les neurones de la couche de sortie

$v$ : potentiel synaptique du neurone

$x$ : entrée du réseau

Pour les couches cachées le choix de la fonction réside dans sa dérivabilité et sa simplicité de calcul pour l'apprentissage. Nous retenons la fonction tangente hyperbolique, sigmoïde ramenée entre –1 et 1 :

$$Sig(v_i) = \frac{e^{2v_i} - 1}{e^{2v_i} + 1}$$

L'apprentissage est une étape indispensable pour pouvoir discriminer les classes et être capable de généraliser le problème avec un nombre limité d'échantillons en apprentissage. En effet, dans un contexte d'écriture manuscrite, il n'est pas possible de récupérer tous les prototypes possibles. Pour réaliser l'apprentissage il faut posséder une base de données,

suffisamment importante par rapport au nombre de poids du PMC. L'apprentissage se fait en règle générale de manière supervisée, c'est-à-dire que la classe de chaque donnée est connue de façon certaine et est utilisée pour l'apprentissage.

#### 4.3.2.2. Algorithme d'apprentissage

L'apprentissage est basé sur la correction des erreurs de classement des données. Pendant l'entraînement supervisé, le système modifie graduellement ses poids pour que sa sortie tende vers la sortie désirée.

Supposons :

- $X_e$, le vecteur d'entrée
- e, l'indice de l'échantillon courant de la base d'apprentissage
- w, le vecteur de paramètres du réseau (les poids),
- $Y(X_e,w)$ le vecteur de sortie
- et $Y_d(X_e)$ le vecteur de sortie désiré.

Apprendre les exemples de la base de données revient à minimiser l'erreur globale sur toute la base définie par l'équation ci-dessous :

Erreur locale : $d(Y(X_e,W),Y_d(X_e))$ ; (l'erreur sur un échantillon)

Erreur globale : $E(W)=\sum_e d(Y(X_e,W),Y_d(X_e))$ ; (l'erreur sur toute la base )

En général, on utilise principalement pour adapter les poids les méthodes utilisant le gradient, celles-ci sont très efficaces. Les algorithmes de gradient conjugué sont nombreux mais coûteux en temps, calculs et mémoires. On préfère la méthode de descente de gradient qui est une méthode itérative et dont la règle de mise à jour est la suivante :

- initialisation des poids $W^{(0)}$
- règle de mis à jour : $W_{l,j,i}^{n+1}=W_{l,j,i}^{n}-\mu\times\nabla E(W_{l,j,i}^{n})$ ; $\mu$ est le pas de descente, $l$ l'indice de couche et $i$ l'indice du neurone. Le gradient global $\nabla E(W_{l,j,i}^{n})$ est la somme des gradients locaux : $\nabla E(W_{l,j,i}^{n})=\sum_e \nabla E_{loc_e}(W_{l,j,i}^{n})$ où :

$$\nabla E_{loc_e}(W_{l,j,i}^{n})=\frac{\partial d(Y(X_e,W^n),Yd(X_e))}{\partial w_{l,j,i}^{n}}$$

$$=\frac{\partial d(Y(X_e,W^n),Yd(X_e))}{\partial Y(X_e,W^n)}\frac{\partial Y(X_e,W^n)}{\partial w_{l,j,i}^{n}}$$

La méthode stochastique, soit par optimisation de l'erreur par le gradient local, entraîne bien la convergence, comme la méthode classique. Certains auteurs comme Yann LeCun ont montré dans la pratique que la méthode stochastique est bien plus rapide que le gradient global [**114**]. La rétro propagation est très efficace pour calculer le gradient de la fonction de l'erreur d'un perceptron multi couche, comparée à la méthode directe [**115**]. Nous allons utiliser cette méthode par la suite dans nos systèmes[9].

L'algorithme d'apprentissage se fait donc en trois passages :

- Le premier passage propage l'entrée pour connaître en sortie l'estimation du réseau, on obtient un vecteur Y de sortie pour l'entrée $X_e$.
- Le second passage, de la sortie vers l'entrée, permet de calculer les erreurs commises par chaque neurone en commençant par la couche de sortie et en remontant vers la couche d'entrée.
- Le troisième suit le sens de la propagation pour mettre à jour les poids du réseau.

#### 4.3.2.3. Paramétrage du réseau

La rétropropagation peut être particulièrement lente pour des réseaux multicouches. Aucune formule ne garantit la convergence du réseau vers l'optimum global. Cependant, il est possible d'augmenter les chances du réseau de converger en paramétrant plusieurs facteurs tels que :

- normalisation des entrées,
- initialisation des poids,
- fonction de transfert de type sigmoïde,
- apprentissage global ou stochastique…

Le pas de gradient, dans la correction des poids, est aussi un facteur déterminant dans l'apprentissage d'un réseau de neurones. Il peut être ajusté automatiquement : on diminue le pas de gradient lorsque la matrice des poids oscille et à l'inverse on l'augmente lorsque celle-ci suit toujours la même direction. Une solution pour utiliser un pas approprié aux données est de régler le pas de manière cyclique [**116**]. Pour commencer, on choisit un pas relativement fort pour accélérer la convergence, ensuite on diminue progressivement sa valeur pour affiner la solution, puis on réitère ce schéma, quitte à perdre temporairement en précision, afin d'éviter de tomber dans un minimum local.

[9] Une description complète des calculs de gradient se trouve dans [**63**]

#### 4.3.2.4. Architecture du réseau de neurones

La capacité du réseau de neurones dépend du nombre de paramètres libres (poids du réseau). Pour être capable d'apprendre des surfaces de décision complexe, le PMC doit avoir un nombre de couches suffisant (un minimum d'une couche cachée) et un nombre suffisant de neurones sur cette (ces) couche(s) cachée(s). Cependant, la qualité des résultats dépend beaucoup de la représentativité de l'ensemble d'apprentissage et des caractéristiques d'entrée extraites sur les données.

Il n'y a pas une structure unique ou optimale d'un réseau. On peut obtenir la même performance avec des topologies différentes de réseaux, en termes de taille et type de connexions. Mais le choix de la configuration du réseau −initialisation des poids, choix de la fonction d'activation, nombre de neurones, nombre de couches cachées− reste crucial pour pouvoir faire converger le réseau vers une performance optimale.

La structure du réseau peut être choisie de façon empirique. Par l'expérience, on compare les performances obtenues pour différentes architectures, puis on conserve celle correspondant au meilleur compromis performance/coût architectural. La taille de la couche d'entrée est souvent choisie égale à la taille du vecteur de caractéristiques. Le nombre de neurones de la couche de sorties est égal au nombre de classes que l'on cherche à distinguer avec une possibilité d'un neurone supplémentaire pour qualifier les éléments appartenant à aucune des classes, appelé classe de rejet [**116**][**117**] ou classe « nil » (classe de non présence de caractères) dans [**118**]. Le nombre de couches cachées et de neurones associés varient selon la précision souhaitée. L'intérêt d'une optimisation empirique réside dans la possibilité d'intégrer des connaissances préalables des données comme par exemple, l'intégration de connaissances temporelles ou spatiales pour les réseaux à convolution que nous présentons à la section suivante.

D'autre part, la structure du réseau peut être optimisée en utilisant des algorithmes automatiques. On peut citer les algorithmes de croissance et de dégénérescence de réseaux de neurones. Comme son nom l'indique le but de l'algorithme de croissance est d'effecteur l'apprentissage avec au départ une architecture très petite et de l'accroître en fonction des performances obtenues. Les algorithmes de dégénérescence à l'inverse partent d'un réseau surdimensionné que l'on simplifie au cours de l'apprentissage par diminution du nombre de neurones et de connexions. Nous n'approfondirons pas à ces méthodes, nous pouvons citer les travaux de Torres-Moreno dont la thèse [**119**] expose les différentes méthodes de calculs des capacités d'un réseau de neurones et propose des algorithmes constructifs.

Une autre approche, le méta-apprentissage, est une passerelle entre la méthode empirique et la méthode d'optimisation automatique. Le méta-apprentissage consiste à apprendre à apprendre : on utilise par exemple un

méta-réseau de neurones pour apprendre à configurer un réseau de neurones classique, qui lui s'occupe d'un problème concret. Le méta-réseau est un réseau de neurones qui apprend à reconnaître une bonne configuration (sans la tester) afin d'optimiser les tirages aléatoires. En entrée du méta-réseau on indique les configurations testées, et en sortie les scores obtenus correspondants. Ensuite, seules les configurations (toujours tirées au hasard) que le méta-réseau juge intéressantes seront réellement testées [**120**][**121**].

#### 4.3.2.5. Réseaux de neurones à convolution (TDNN : Time Delayed Neural Network)

La principale différence entre les réseaux de neurones classiques de type perceptron et les réseaux de neurones à convolution réside dans leur architecture et plus précisément dans leur approche connexionniste. La Figure 57 illustre cette différence. Un neurone d'une couche d'un perceptron est connecté à tous les neurones de la couche précédente (y compris la couche d'entrée) tandis que pour un réseau de neurones à convolution, un neurone est connecté à un sous-ensemble de neurones de la couche précédente. Chaque neurone peut être vu comme une unité de détection d'une caractéristique locale.

Les réseaux de neurones à convolution de type TDNN incorporent des contraintes et réalisent un certain degré d'invariance de décalage et de déformation en utilisant deux idées : zones réceptives locales et poids partagés. L'utilisation des poids partagés réduit le nombre de paramètres dans le système facilitant la généralisation. Ce type de réseau a été avec succès appliqué à la reconnaissance des chiffres [**122**].

Le concept de poids partagés [**123**][**124**] se base sur le principe que dans le cerveau humain des neurones détectent certains traits dans de petites régions de la rétine, essentiellement de la même manière dans toutes ces régions. On a ainsi plusieurs neurones qui calculent la même fonction sur des entrées différentes. Ainsi découle le fait qu'il est nécessaire qu'ils partagent les mêmes poids.

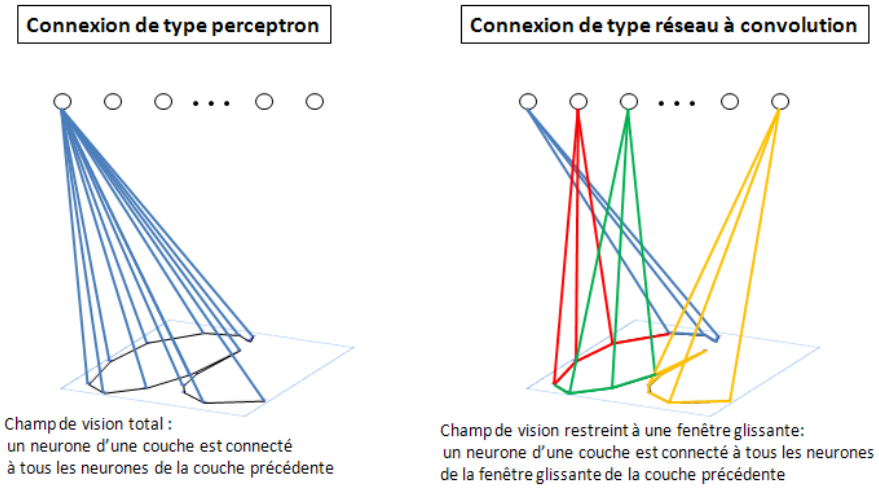



Figure 57 - Illustrations des connexions dans un PMC et dans un TDNN

Le TDNN est un réseau à délai utilisé pour des données de nature séquentielle, donc adapté pour la reconnaissance de l'écriture en-ligne. La topologie du TDNN se caractérise par deux parties. La première, correspondant aux couches basses, implémente les convolutions successives permettant de transformer progressivement une séquence de vecteurs caractéristiques en une autre séquence de vecteurs caractéristiques d'ordre supérieur. La seconde correspond à un PMC classique, elle reçoit en entrée l'ensemble des sorties de la partie extraction.

Nous avons adopté un PMC à une couche cachée de 100 neurones, et de 350 neurones d'entrée corresponds au vecteur de caractéristique. En ce que concerne le TDNN nous avons repris la topologie optimale proposée dans [**74**] illustrée dans la Figure 58.

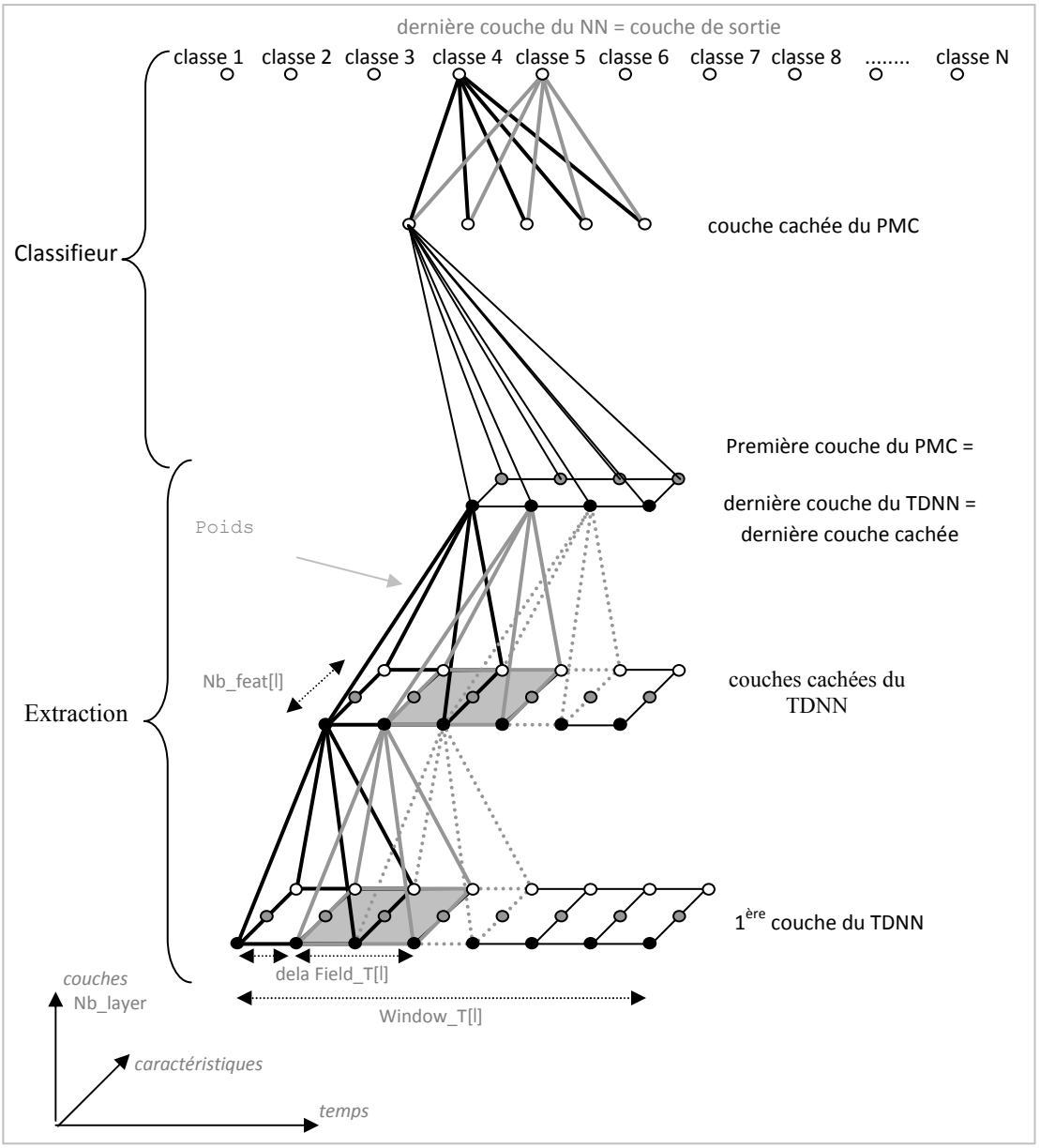



Figure 58 - Architecture du TDNN [**74**]

La première couche du réseau acquiert les caractéristiques du signal. Une couche cachée du réseau de neurones (phase d'extraction) transforment une séquence de vecteurs caractéristiques en une autre séquence de vecteurs caractéristiques d'ordre supérieur. Un neurone donné détecte une caractéristique topologique locale de la trajectoire du stylo. Le champ de vision du neurone est restreint à une fenêtre temporelle limitée. Avec la contrainte des poids partagés, le même neurone est dupliqué dans la direction temps (soit la même matrice des poids dupliquée) pour détecter la présence ou

l'absence de la même caractéristique à différentes places le long de la trajectoire du signal. En utilisant plusieurs neurones (nb_feat) à chaque position temporelle, le réseau de neurones effectue la détection de caractéristiques différentes : les sorties des différents neurones produisent un nouveau vecteur caractéristique pour la couche supérieure.

Enfin, la première couche de la partie classifieur (PMC entièrement connecté) correspond à la dernière couche de la partie extraction. En sortie, le PMC donne les probabilités de l'entrée pour chaque classe définie.

### 4.3.3. Différentes architecture de classifieurs

Nous revenons ici sur la notion de rejet pour expliciter différentes architectures que nous allons proposer. Un classifieur de symboles peut être employé dans différents contextes. Classiquement, il est utilisé dans un contexte de symboles isolés. Dans ce cas, l'entrée est bien un symbole connu, et on attend du classifieur qu'il associe la bonne classe (ou la classe la plus probable) à cette entrée. Le deuxième contexte qui nous intéresse plus est celui d'un classifieur implanté dans un système global de reconnaissance d'expressions mathématiques. Le rôle du classifieur n'est plus seulement d'associer l'entrée à une classe, mais d'établir un (des) score(s) de reconnaissance à chaque hypothèse de segmentation. Dans ce contexte, le classifieur doit avoir la capacité d'identifier les mauvaises segmentations et leur donner des coûts élevés. Autrement dit, nous sommes face au problème de rejet. Parmi plusieurs méthodes existantes de rejet nous avons exploré la possibilité d'utiliser un classifieur hybride comportant en cascade un classifieur de rejet (***classifieur rejet***) qui décide si l'hypothèse est acceptée ou rejetée et le classifieur principal (***classifieur cible***) qui lui décide de sa classe [**125**]. Une autre façon de faire est de considérer les mauvaises hypothèses comme appartenant à une classe spécifique appelée la classe de rejet [**126**]. Cette classe est alors ajoutée au problème de classification et il faudra en tenir compte pendant l'apprentissage.

#### 4.3.3.1. Classifieur de symboles isolés sans rejet

Dans ce cas le classifieur utilisé est un réseau de neurones classique. Que ce soit un PMC ou un TDNN, le nombre de sorties du classifieur est égal au nombre de classes des symboles. L'avantage d'utiliser ce classifieur est qu'on peut l'entraîner sur une base de symboles isolés. Cet apprentissage est rapide et efficace. Il est possible d'obtenir un bon classifieur dans un contexte isolé. Toutefois, ce classifieur est imprédictible face aux segmentations invalides des traits. Cela montre l'intérêt d'introduire la classe de rejet au classifieur pour pouvoir identifier ces mauvaises segmentations.

#### 4.3.3.2. Classifieur hybride avec rejet

Pour faire face au problème d'identification de rejet, nous proposons un classifieur hybride. Le classifieur est découpé en deux classifieurs spécialisés,

qui coopèrent comme montré Figure 59. Il est constitué d'un classifieur cible destiné à reconnaître les symboles valides. Celui-ci est appris soit en isolé, lorsque le classifieur ne supporte pas un apprentissage incrémental –cas du SVM par exemple–, soit il peut être également appris globalement dans l'architecture du système –cas du réseau de neurones–. Ce classifieur va donc participer à l'apprentissage global du second classifieur, dit classifieur de rejet.

En ce qui le concerne, le classifieur de rejet a comme objectif la reconnaissance de deux classes, rejet et non-rejet (symboles valides). Ce classifieur est appris globalement puisque les exemples de rejet sont obtenus au cours de l'étape de segmentation d'expressions. On peut penser que ce classifieur soit plus simple à apprendre en mode global (deux classes seulement) qu'un classifieur complet (N+1 classes).

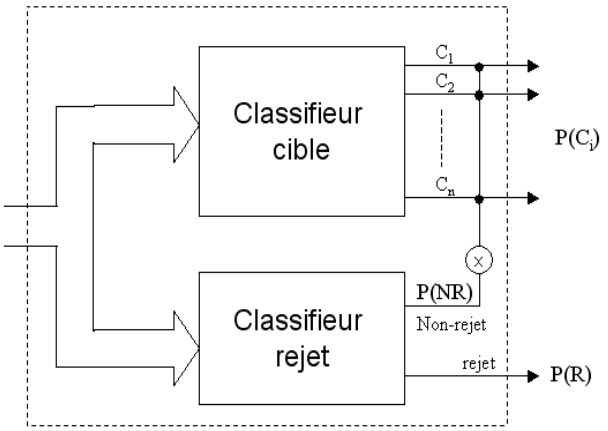



Figure 59 - Le classifieur hybride

Pour pouvoir combiner les sorties des ces classifieurs, il importe que leurs sorties soient normalisées. Cette normalisation est réalisée en appliquant une fonction de type « softmax » sur les sorties des classifieurs [**127**]. Ainsi, les sorties de chaque classifieur sont en forme de probabilités indépendantes et l'on peut montrer ci-dessous que ce schéma produit bien les $N$ probabilités $P(C_i)$ d'une part et la probabilité $P(rejet)$ d'autre part.

La probabilité qu'une sortie du classifieur cible soit la classe $C_i$, sachant que ce n'est pas un rejet (NR) est :

$p(C_i \mid \mathrm{NR})$ ; avec $\sum_{i=1}^{N} p(C_i \mid \mathrm{NR}) = 1$ où N est le nombre de symboles.

Par ailleurs, la sortie du classifieur rejet nous donne la probabilité qu'une hypothèse soit une classe à accepter (NR) ou à rejeter (R) :

$$p(NR) + p(R) = 1$$

Mais la probabilité qu'une sortie du classifieur hybride soit la classe $C_i$ est :

$$p(C_i) = p(C_i, NR) + p(C_i, R) = p(C_i \mid \text{NR}).\text{p(NR)} + p(C_i \mid \text{R}).\text{p(R)}$$

Mais par définition: $p(C_i \mid \text{R}) = 0$, donc :

$$p(C_i) = p(C_i \mid \text{NR}).\text{p(NR)} \tag{11}$$

Cette probabilité est normalisée par rapport à celle du rejet permettant donc de combiner directement le rejet avec les classes du classifieur cible, car :

$$\sum_{i=1}^{N} p(C_i) + p(R) = 1 \tag{12}$$

#### 4.3.3.3. Classifieur global avec rejet

La solution d'utiliser un classifieur spécialisé dans la reconnaissance des formes à rejeter semble intuitive. Une autre solution, plus compacte, est de fusionner ces deux classifieurs en un seul en intégrant la classe de rejet comme N+1$^{\text{ème}}$ sortie du classifieur. A priori le problème posé est plus complexe puisque un même classifieur doit modéliser à la fois chacune des classes existantes et la fameuse classe correspondant aux segmentations invalides. Dans ce cas, le rejet est considéré comme une classe comme les autres pendant l'apprentissage et la reconnaissance. Il faudra pour cela disposer d'exemples de cette classe additionnelle lors de l'apprentissage global du système. La probabilité est donc obtenue directement en sortie du classifieur.

Après avoir obtenu le score de reconnaissance d'une hypothèse en forme de probabilité, le coût d'association de l'hypothèse courante à une classe *j* est donnée par :

$$C_{reco}(hs_i) = \begin{cases} Coût\_max \text{ si } \text{C}_\text{j} = rejet \\ -log(P(c=C_j \mid hs_i)); autrement \end{cases} \tag{13}$$

En donnant un coût maximal aux hypothèses reconnues comme rejet, on empêche le bloc de décision de choisir les solutions incluant ces hypothèses (si aucune analyse syntaxique n'interdit l'utilisation de cette classe dans le résultat).

Une autre stratégie consiste à utiliser le complément de la probabilité au lieu d'exclure directement les hypothèses invalides. Cela donne plus de flexibilité surtout pendant l'apprentissage du classifieur. Dans ce cas la formule de coût devient :

$$C_{reco}(hs_i) = \begin{cases} -log(1 - P(c=rejet \mid hs_i)) \\ -log(P(c=C_j \mid hs_i)); \text{C}_\text{j} \neq rejet \end{cases} \tag{14}$$

Nous allons dans le paragraphe suivant présenter la philosophie de l'apprentissage global et les stratégies employées pour y parvenir.

## 4.4. Apprentissage global du classifieur

Le point clé des classifieurs proposés ci-dessus est leur capacité à identifier le rejet. Pourtant, il n'est pas évident de savoir comment apprendre cette classe à partir de symboles isolés, et cela pour plusieurs raisons. Premièrement, on ne possède pas intrinsèquement de base d'exemples « rejet » car par définition, le rejet représente « Autre chose ». Cet « Autre chose » vient de sous parties de symboles, de regroupements de sous parties de symboles, ou même de regroupements de symboles complets. La constitution d'une telle base est envisageable en synthétisant ces regroupements à partir d'une base isolée. Pour cela, il est possible d'utiliser le générateur d'hypothèses en sauvegardant toutes les hypothèses invalides. Malheureusement, cette dernière solution n'est pas très pratique car le nombre d'exemples de rejet sera immense. Par exemple, pour une simple expression de 4 symboles composés en 7 traits, il y a 873 regroupements invalides, ce nombre augmente rapidement en fonction de nombre de traits, (selon le nombre de Bell cf. chapitre 2). En conséquence, le nombre d'exemples obtenus d'une base d'un millier d'expressions serait de l'ordre du million d'exemples de rejet. Conserver une sous partie de cette base peut être une solution, mais en pratique il est difficile de choisir les exemples qui sont les plus appropriés pour représenter au mieux la classe de rejet.

D'où l'idée de ***l'apprentissage global*** du classifieur qui s'effectue grâce aux résultats courants de la reconnaissance d'expressions. Le système est entraîné sur la base d'apprentissage d'expressions complètes, soit sans tenir compte de la grammaire (modèle libre), soit en en tenant compte (modèle contraint). L'objectif de cette phase est l'apprentissage du classifieur de symboles, y compris la classe de rejet. Cet apprentissage s'appuie sur la vérité terrain de l'expression courante E et le résultat de reconnaissance E' renvoyé par le reconnaisseur d'expressions dans son état actuel d'apprentissage. Le classifieur est mis à jour selon l'algorithme suivant :

Algorithme 1 - Apprentissage global du classifieur

**Pour** toutes les expressions d'apprentissage E
    E' = résultat de reconnaissance
    **Pour** tous les symboles $s'_i$ de E'
        // Cas des symboles bien segmentés
        **Si** les traits de $s'_i$ correspondent à un symbole $s_i$ de la vérité E
            **Si** ($s'_i = s_i$) **Alors** ne rien faire // symbole bien reconnu
                **Sinon** apprendre $s_i$ // symbole mal reconnu
        // Cas des symboles mal segmentés
        **Si** les traits de $s'_i$ ne correspondent à aucun symbole de la vérité E
            **Alors** apprendre $s'_i$ en tant que rejet

    **Pour** tous les symboles $s_i$ de E non trouvés dans E'
        apprendre $s_i$

La mise à jour du classifieur consiste donc à apprendre tous les exemples renvoyés par l'algorithme pour chaque expression. En conséquence, le classifieur doit avoir la capacité d'apprentissage itératif où chaque nouvel exemple vient mettre à jour l'état actuel du classifieur. Les réseaux de neurones sont très appropriés pour cette tâche. Par contre, une telle stratégie n'est pas envisageable avec un classifieur de type SVM classique. L'algorithme ci-dessus se répète jusqu'à la convergence du classifieur. La Figure 60 illustre un exemple de mise à jour requise dans les trois cas montrés dans l'algorithme. Dans le premier cas, les traits du symbole sont bien regroupés ensemble, et définissent donc une bonne segmentation. C'est le cas des symboles ('5', '=', '8'). Puisque, le '5' et le '=' ont été bien reconnu ils ne participeront pas à l'apprentissage. Par contre, le segment du '8' a été mal reconnu (reconnu en tant que '6'), donc il doit contribuer à une évolution du classifieur pour favoriser la reconnaissance d'un '8'.

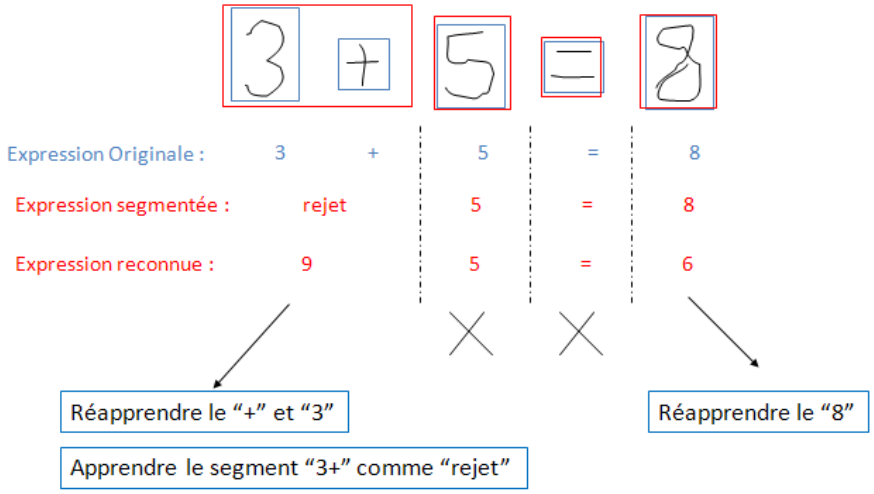



Figure 60 - Les mis à jour nécessaires au cours de l'apprentissage global sur l'expression 3+5=8

De même, la segmentation qui regroupe les traits de '3' et '+' n'a pas de correspondance dans la vérité terrain et représente donc une mauvaise segmentation que le classifieur devrait classer dans la classe rejet. Or, dans cet exemple, ce n'est pas le cas, cet élément étant reconnu comme un '9'. Il en résulte un exemple qui va être utilisé pour mettre à jour l'apprentissage et favoriser sa reconnaissance comme rejet. Finalement, les segments correspondant au '3' et au '+' n'apparaissent pas dans la solution, ils sont donc également réappris associés à leur classe respective. Chaque réapprentissage entraîne une mise à jour du classifieur.

En ce qui concerne le classifieur hybride la même méthodologie est suivie mais en ne mettant à jour que le classifieur de rejet, car on suppose que le classifieur cible est déjà appris de façon optimum. Prenons l'exemple de la

Figure 61, et supposons que l'expression « x – 1 » soit segmentée et reconnue comme « x1 » à cause d'une mauvaise segmentation regroupant les traits du 'x' et du '–'. Cette mauvaise segmentation va entraîner trois mises à jour du classifieur de rejet/non-rejet. Premièrement, l'ensemble de traits correspondant à la mauvaise segmentation est appris en tant que classe rejet. Puis, les traits du 'x' sont appris en tant que non-rejet (vrai symbole), de même pour le trait du '–'. Ce processus est répété sur toute la base d'apprentissage d'expressions complètes jusqu'à convergence de l'apprentissage du classifieur de rejet.

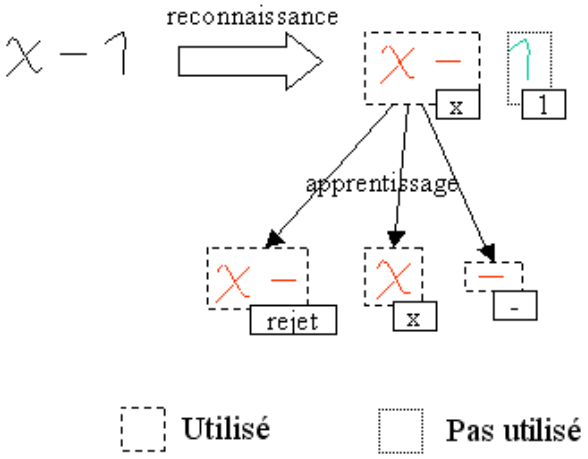



Figure 61 - Illustration de l'apprentissage du classifieur « rejet »

Après avoir introduit cette notion d'apprentissage global qui inclut l'apprentissage au sein de l'architecture pour tenir compte des propriétés du segmenteur d'expressions, il est possible décliner plusieurs stratégies, illustrées dans la Figure 62, pour réaliser cet apprentissage global :

***Apprentissage global pur***

Dans ce cas, le classifieur est initialisé aléatoirement (classifieur vide). Puis la base d'apprentissage d'expressions est utilisée dans une boucle d'apprentissage global pour entraîner le classifieur.

***Apprentissage iso-global***

Avant de faire l'apprentissage global, le classifieur est initialisé en faisant un apprentissage sur la base isolée, dans le but de mieux apprendre les classes peu fréquentes dans la base d'expressions.

***Apprentissage globo-isolé***

A l'inverse du cas précédent, l'apprentissage isolé est fait après le global pour viser le même objectif.

***Apprentissage isolé en global***

Pour éviter l'apprentissage en deux étapes (isolé – global), la base de symboles isolés peut être utilisée en tant que base d'expressions au cours de

l'apprentissage global. Dans ce cas, chaque symbole est considéré comme une expression constituée d'un seul symbole.

Dans les expérimentations qui vont être présentées dans le chapitre suivant, nous allons voir les avantages et les inconvénients de chacune de ces stratégies. Elles seront également comparées avec une structure classique de classifieur appris en isolé sans capacité de rejet.

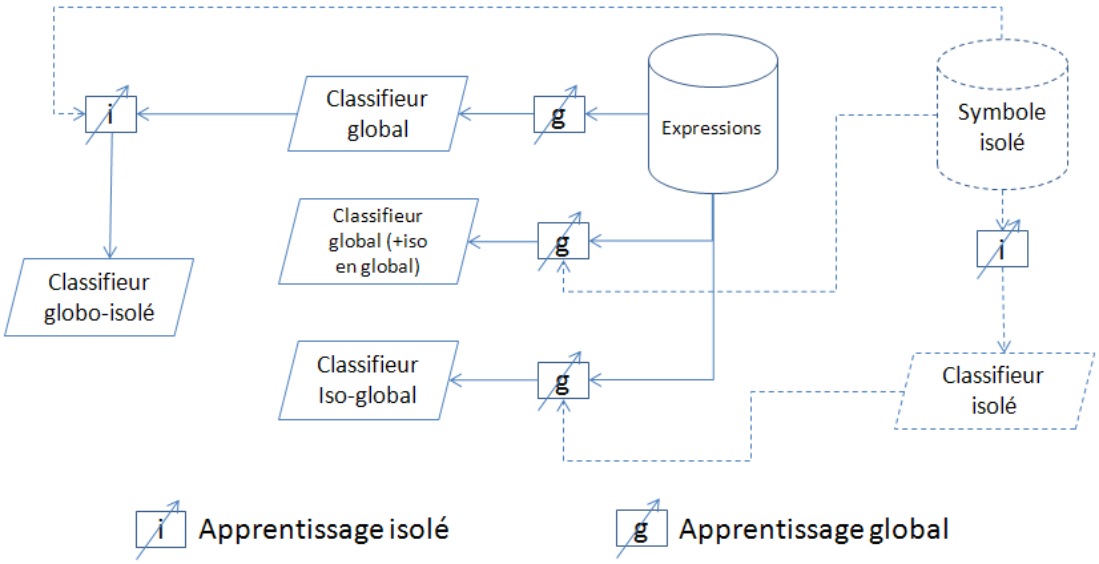



Figure 62 - Méthodologies d'apprentissage du classifieur (isolé et global)

## 4.5. Représentation d'une expression mathématique manuscrite en-ligne

Comme nous l'avons introduit dans le chapitre 2, une expression mathématique peut être représentée par un arbre. Cet arbre est construit à partir d'une grammaire permettant de vérifier la syntaxe de l'expression. Chaque règle de la grammaire permet donc de générer soit des nœuds terminaux (portant les symboles et leur signification) soit des nœuds non-terminaux mettant en relation logique et spatiale les différentes sous parties d'une expression. Pour valuer ces relations et permettre d'identifier celles présentes dans l'expression à reconnaître, nous avons défini des fonctions de coût qui pénalisent plus ou moins les hypothèses selon la position et la taille relatives de leurs constituants par rapport à une référence.

Nous décrivons maintenant le type d'arbre relationnel que nous utiliserons pour représenter une expression candidate. La racine de cet arbre définira la solution proposée ainsi que le coût global de cette solution, le type de relation entre ses nœuds fils, et son étiquette. De la même façon, les nœuds non terminaux représentent des sous-expressions, leurs coûts, la relation qui regroupe ses nœuds fils, et leurs étiquettes. Enfin, les nœuds terminaux sont les hypothèses de symboles proposés par le générateur d'hypothèses avec leurs

coûts de reconnaissance. Prenons un exemple simple, la grammaire définie dans le Tableau 13 génère des arbres candidats similaires à celui de la Figure 63. Les règles qui produisent les nœuds terminaux ne sont associées à aucune relation spatiale. A l'inverse, à chaque règle qui produit un nœud non-terminal un type de relation spatiale est associé. Détaillons maintenant chacun de ces deux types de nœuds.

***Les nœuds non-terminaux (NT) :***

Un nœud de ce type contient une sous-expression SE (décrite par l'ensemble des traits concernés et la chaîne LaTeX correspondante) produite par la combinaison des sous-expressions reliées par la relation spatiale R. Le coût d'un nœud non-terminal (y compris la racine) est le coût, appelé coût structurel, pour que les sous-expressions ($SE_i$) soient reliées par la relation R : $C_{struct}(R|SE)$

***Les nœuds terminaux (T):***

Chacun de ces nœuds contient les traits des hypothèses retenues, celles-ci provenant du générateur d'hypothèses. Une hypothèse est identifiée par une étiquette (résultant du classifieur) et son coût de reconnaissance $C_{reco}(hs_i)$.

Finalement, le coût global d'une expression candidate de *n* symboles reliés par *r* relations est :

$$C_E = \sum_{i=1}^{n} C_{reco}(hs_i) + \sum_{j=1}^{r} \alpha . C_{struct}(R_j | SE_j) \qquad (15)$$

Le facteur α est très important pour compenser la différence (s'il y en a) entre les coûts structurels, et les coûts déduits des scores de reconnaissance d'hypothèses. Il est également choisi expérimentalement comme nous allons voir dans la section 5.4.

Tableau 13 - Exemple d'une grammaire simple

| Règle | Type de relation |
|---|---|
| sym ← x, y, 1, 2, … | |
| op ← +, -, x, … | |
| formule ← subExp op sym | (operator) |
| subExp ← subExp op sym | (operator) |
| subExp ← sym sym | (superscript) |
| subExp ← sym | |

La Figure 63 montre un exemple d'un arbre relationnel. Le nœud non-terminal '2' relie les terminaux '5' et '6' qui contiennent les hypothèses des symboles *x* et *2* respectivement par la relation « superscript » produisant la sous-expression $x^2$. Le deuxième nœud non-terminal est la racine de l'arbre. Il

relie les nœuds '2', '3', et '4' par la relation « operator » produisant le résultat final $x^2$-*1*. Le coût global de cette solution est :

$$C_E = C_{reco}('x') + C_{reco}('2') + C_{reco}('-') + C_{reco}('1') + \alpha.C_{struct}(R_1 | x^2) + \alpha.C_{struct}(R_2 | x^2 - 1)$$

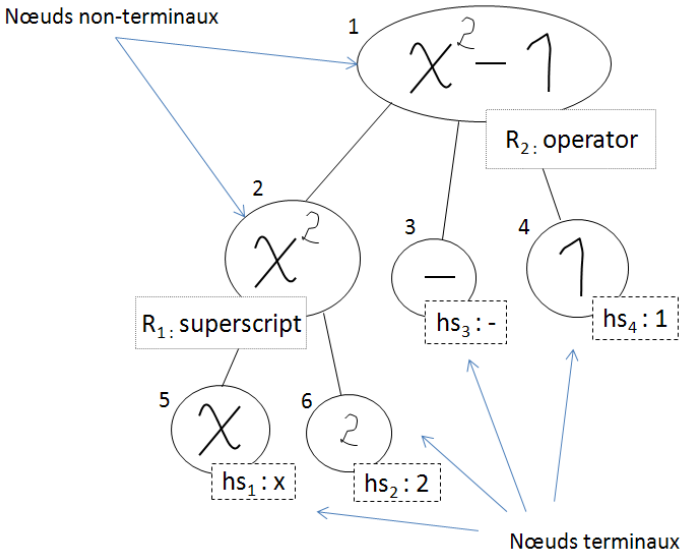



Figure 63 - Exemple d'arbre relationnel

Plusieurs résultats respectant la grammaire peuvent être proposés par le reconnaisseur d'expressions, comme montre la Figure 64.

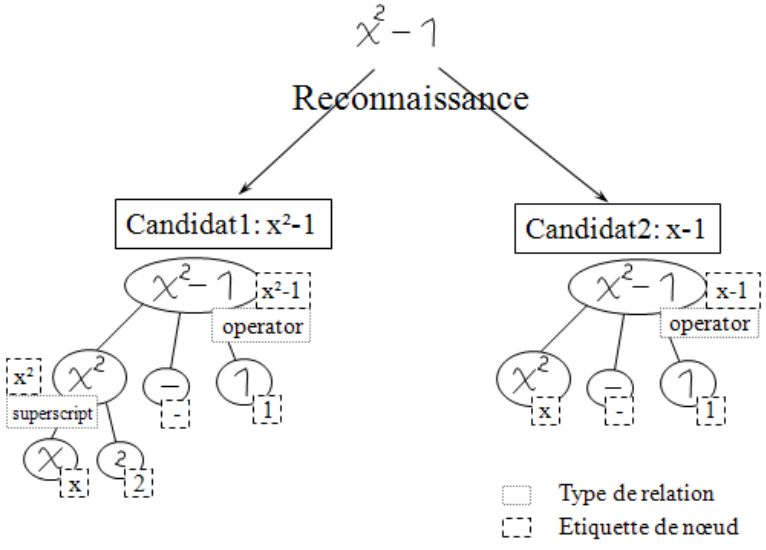



Figure 64 - L'arbre relationnel de deux expressions candidates

## 4.6. Analyse structurelle

L'analyse structurelle requiert l'extraction d'informations spatiales de chacune des hypothèses de symbole afin d'accomplir deux tâches : déduire les

informations spatiales des sous-expressions et effectuer l'analyse syntaxique. Mais, comme nous l'avons vu, il existe des ambiguïtés, pour y pallier nous calculons des informations spécifiques selon le type de symbole. D'une façon générale, l'analyse structurelle se base sur l'alignement et la taille des symboles. Ainsi, la relation qui va prévaloir entre deux symboles (ou deux sous-expressions) est établie à partir de la ligne de base (y) de chaque élément et la taille (h) de chaque élément. La Figure 65 - montre un exemple des informations extraites pour les symboles de type ascendant. Les informations spatiales des autres types de symboles sont détaillées dans l'annexe D-1.

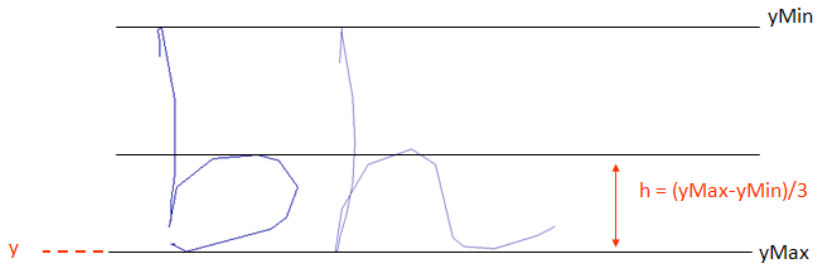



Figure 65 - Les informations structurelles des symboles de type ascendant

De plus, les informations spatiales d'une sous-expression sont calculées à partir de ses éléments en considérant le type de la relation qui produit cette sous-expression. Quand une relation établie une sous-expression SE, les valeurs de sa ligne de base et sa hauteur sont déduites des N composantes produisant cette sous-expression $SE_1,...,SE_N$ :

$$y_{SE} = f_R^y(y_{SE_1},...,y_{SE_N}); h_{SE} = f_R^h(h_{SE_1},...,h_{SE_N})$$

Le nombre N des nœuds fils (sous-expressions) diffère en fonction du type de la relation R ; ici N={2, 3, 4}. Nous introduisons également la notion de sous relation, elle est par définition la relation entre un nœud et un seul de ces nœuds fils. Le Tableau 14 montre toutes les relations et leurs sous-relations considérées par l'analyse structurelle en détaillant leurs représentations structurelles et en arbre. Par exemple, un nœud avec la relation « superscript » a deux fils, où le deuxième est en exposant du premier. Dans l'exemple de la Figure 66 : x et 2 sont les fils de la sous-expression $x^2$ via la relation « superscript » (et les sous-relations « base » et « exposant »), les informations spatiales de la sous-expression produite sont calculées par (HW et YW étant des poids de combinaison) :

$$YW_r = YW_x ; HW_r = HW_x + 0.5HW_2$$

$$h_r = \frac{h_x HW_x + h_2 HW_2}{HW_r} \; ; \; y_r = y_x$$

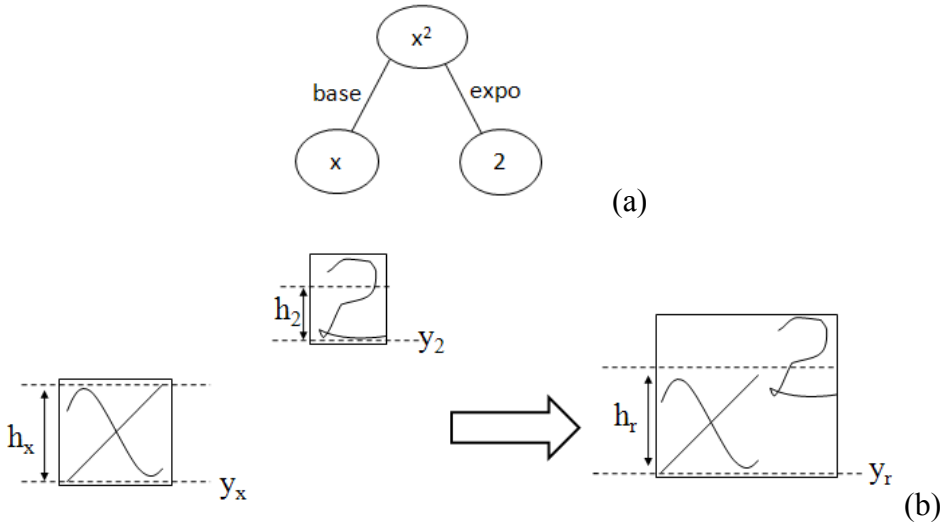
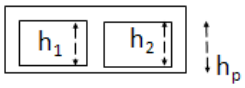
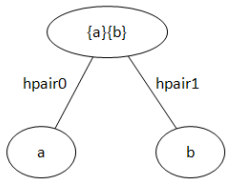
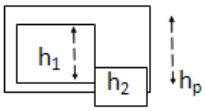
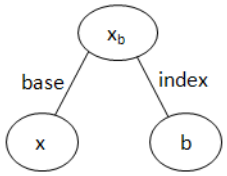
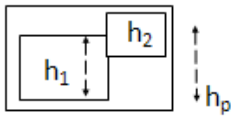
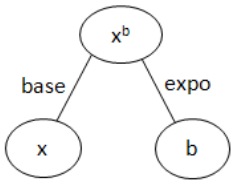
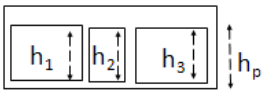
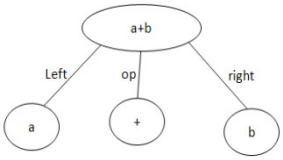
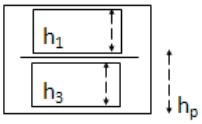
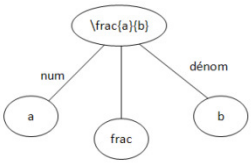
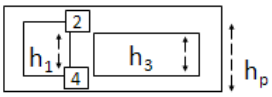
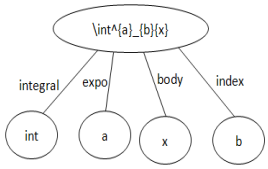
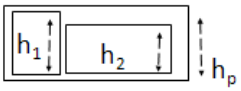
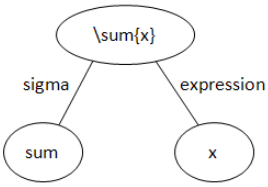
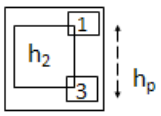
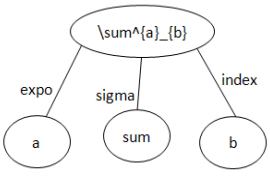
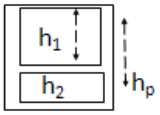
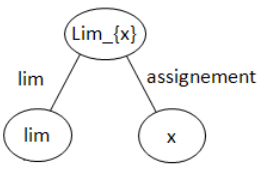


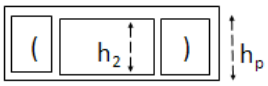
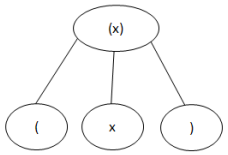
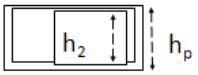
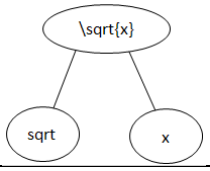

Figure 66 - La relation « superscript » : (a) La représentation en arbre, (b) Les informations spatiales

La notion de poids (HW ,YW) est introduite afin de normaliser les information des sous-expressions selon leurs tailles (nombre des symboles). Initialement, tous les terminaux ont un poids égal à 1. Ensuite, les poids des non-terminaux sont calculés au fur et à mesure selon le type de relation. Ainsi, dans l'exemple précédent, la position et la taille du *2* auront moins d'influence que celles du *x* dans le calcul de la position et la taille de l'expression $x^2$. Chaque relation a sa propre formulation pour calculer ses paramètres $HW_r$, $YW_r$, $h_r$ *et* $y_r$.

Comme nous l'avons introduit dans l'équation (15) un coût structurel est associé à chaque nœud non-terminal, il est calculé en fonction des tailles et positionnements des fils correspondant à la sous-expression de ce nœud. Une solution intuitive est de calculer la différence de positionnement et taille de chaque fils par rapport à une situation idéale définie par des règles empiriques : on obtient ainsi un coût, appelé ***coût géométrique***. Néanmoins, les positionnements idéaux sont très difficiles à définir, à cause de la nature floue des relations spatiales entre les symboles dans les expressions manuscrites. En conséquence, nous essayerons de modéliser (apprendre) ces relations au lieu de définir les positions des symboles dans chaque relation par une règle empirique. Les relations apprises sont modélisées par des histogrammes, ou des fonctions gaussiennes pour en déduire des ***coûts probabilistes***.

Tableau 14 - Les relations spatiales considérées dans l'analyse structurelle

| Relation | Sous-Relations | Représentation structurelle | Représentation en arbre |
|---|---|---|---|
| HPair | Hpair_left<br>Hpair_right |  |  |
| Subscript | Sub_base<br>Sub_index |  |  |
| Superscript | Sup_base<br>Sup_expo |  |  |
| Operator | Operator_ left<br>Operator_ op<br>Operator_ right |  |  |
| Fraction | Frac_num<br>Frac<br>Frac_denom |  |  |
| Integral | Int_intergral<br>Int_expo<br>Int_ind<br>Int_ expression |  |  |
| Sigma | Sigma_ sigma<br>Sigma_ expression |  |  |
| Sum | Sum_expo<br>Sum_sigma<br>Sum_index |  |  |
| Lim | Lim_lim<br>Lim_ assignement |  |  |

| | | | |
|---|---|---|---|
| Parenthesis | |  |  |
| Sqrt | |  |  |

### 4.6.1. Coût géométrique

Le coût géométrique est basé sur l'erreur quadratique de positionnements et de tailles des fils par rapport au nœud parent. Pour cela, pour ce qui concerne le positionnement en ordonnée, on définit le décalage attendu *t* entre un nœud fils (c) et son nœud parent (p). Cette grandeur est normalisée par la hauteur du nœud père. On obtient alors à partir des positions et tailles idéales $\hat{y}$ et $\hat{h}$ :

$$t = \frac{\hat{y}_c - \hat{y}_p}{\hat{h}_p}$$

On calcule alors la différence entre la valeur attendue et la valeur observée, soit :

$$dy_{p,c,t} = t.h_p - \left(y_c - y_p\right)$$

De même, on définit un ratio attendu *s* entre les hauteurs d'un nœud fils et d'un nœud père :

$$s = \frac{\hat{h}_c}{\hat{h}_p}$$

et l'on calcule la différence entre la valeur attendue et la valeur observée :

$$dh_{p,c,s} = h_p - h_c / s$$

On en déduit ensuite le coût quadratique pondéré d'alignement de deux éléments : $sy_{p,c,t} = dy^2_{p,c,t}.YW_c$ ; et celui de différence de taille : $sh_{p,c,s} = dh^2_{p,c,s}.HW_c$.

Les valeurs de $sy_{p,c,t_c}$ et $sh_{p,c,s}$ sont nulles lorsque les éléments ont la ligne de base idéale *y* et la taille idéale *h* définies par les paramètres *t* et *s*. Nous allons présenter dans la suite les situations idéales attendues pour chacune des relations et les coûts géométriques déduits à partir de ces situations. Les valeurs de décalage et de ratio de taille ont été fixées expérimentalement. Dans ces formules, on choisit $t > 0$ lorsque le nœud fils doit être descendu par

rapport au nœud père et $t < 0$ lorsqu'il doit se situer plus haut, de même $s < 1$ lorsque le nœud fils est de plus petite taille que le nœud père, et $s > 1$ lorsque le nœud fils doit être plus grand que le nœud père.

Revenant à l'exemple de la relation « superscript », idéalement l'exposant (fils 2) doit être en dessus de son nœud parent avec un décalage de $t = -1.7$, en ayant la moitié de sa taille, $s = 0.5$ :

$$\left.\begin{aligned} C_Y &= (sy_{p,1,0} + sy_{p,2,-1.7}) \\ C_H &= (sh_{p,1,1} + sh_{p,2,0.5}) \end{aligned}\right\} \rightarrow C_{Expo} = \beta.C_Y + \gamma.C_H \quad (16)$$

Les coûts des autres relations sont calculés d'une façon similaire avec des valeurs différentes de $t$ et $s$. Les facteurs $\gamma$ et $\beta$ qui apparaissent dans les formules correspondent au facteur α de l'équation (15).

Quant aux relations « Parenthèses » et « Racine » les coûts structurels ne sont pas utilisés. Pour les parenthèses il suffit de faire la vérification de l'ouverture et la fermeture lors de l'analyse syntaxique. De même quand une racine carrée est rencontrée, on vérifie si l'expression est bien contenue à l'intérieur avec le moins possible de dépassement.

Cette façon de procéder présente l'avantage de ne nécessiter qu'un nombre limité de paramètres ($s$ et $t$) pour définir les modèles relationnels. Ceux-ci sont ensuite fixés de manière empirique à partir de considérations intuitives. On peut par contre regretter que ce coût géométrique représente une erreur quadratique, ce qui rendra sa combinaison avec les coûts probabilistes de reconnaissance problématique (plus le scripteur écrit grand, plus le coût géométrique est grand !), le facteur $\alpha$ de l'équation (15) servant à contrôler la combinaison avec les coûts de reconnaissance.

### 4.6.2. Coût probabiliste

Nous proposons d'utiliser des coûts calculés à partir des probabilités des positions en ordonnées et des tailles des composantes intervenant dans les relations spatiales. Les probabilités sont estimées directement à partir des données de la base d'apprentissage d'expressions mathématiques. On modélise donc des situations réelles au lieu de définir les cas idéaux de ces situations avec des valeurs empiriques.

Nous définissons les différences de positionnement et taille d'une sous-expression $SE_i$ par rapport à son parent $SE$ comme suit :

$$dh_i = (h_{SE} - h_{SE_i}) / h_{SE} \quad (17)$$

$$dy_i = (y_{SE} - y_{SE_i}) / h_{SE} \quad (18)$$

Les différences ($dh$, $dy$) sont donc les écarts normalisés de position et taille de chaque nœud fils par rapport à la sous-expression (donc indépendant de

l'échelle de l'expression). Les distributions des valeurs de *dh* et *dy* pour chaque élément d'une relation (sous-relation) sont modélisées soit par les histogrammes des occurrences, ou par des modèles gaussiens.

#### 4.6.2.1. Apprentissage des modèles des relations

L'apprentissage des modèles de relations se fait en un seul passage de la base d'expressions et sans tenir compte du classifieur. En forçant la bonne reconnaissance des expressions (à partir de la vérité terrain), nous obtenons l'arbre relationnel correcte de chaque expression, il sert ensuite à trouver les occurrences de *dh* et *dy* de chaque sous relation. L'Algorithme 2 résume l'étape de construction des histogrammes. La fonction récursive « verifyRules » prend en entrée la racine de l'arbre, on calcule et stocke *dh*, *dy* des sous relations de la racine par la fonction « dh/y_brute ». Ce processus se répète pour tous les nœuds non-terminaux de l'arbre de la vérité terrain de toute la base d'apprentissage, obtenant ainsi en sortie les deux listes des occurrences des *dh* et *dy* de chaque sous relation.

Algorithme 2 - Parcours de l'arbre relationel pour trouver les occurences de *dh* et *dy*

**float dh_brute(object,fils)**

return $\frac{h_{object} - h_{fils}}{h_{objet}}$

**float dy_brute(object,fils)**

return $\frac{y_{object} - y_{fils}}{h_{objet}}$

**verifyRules**

entrée : obj - l'arbre relationnel de la vérité terrain d'une expression

sortie: mettre à jour les listes de dh, dy pour chaque sous-relation rencontrée dans l'arbre

```
verifyRules (obj);
pour chaque fils de (obj) : f
    sousRel = la sous relation qui relie f et son obj parent
    Si  f a un histogramme // nœud non-terminal
            dh = dh_brute(obj,f);
            dy = dy_brute(obj,f);
            RelationHistogram[sousRel].dhValues.add(dh);
            RelationHistogram[sousRel].dyValues.add(dy);

    Si f est un nœud non-terminal
            verifyRules(child(obj);
```

Les histogrammes des différences de positions et de tailles de symboles pour chaque sous-expression intervenant dans la relation sont construits en quantifiant les valeurs des *dh* et *dy* en N valeurs

Les histogrammes sont sauvegardés en fichiers textes, où pour chacun des histogrammes on sauvegarde les N valeurs de fréquence, et les valeurs minimum et maximum.

De même, nous pouvons utiliser ces listes des occurrences *dh*, *dy* pour construire une fonction gaussienne pour chaque sous relation ( $R^s$ ). Pour les *n* occurrences de *dh* et *dy* on calcule la moyenne μ est la variance de chaque modèle comme suit (i=1..n) :

$$\mu_{SE,dx}^{R^s} = \frac{\sum_i dx_i}{n};$$

$$\sigma_{SE,dx}^{R^s} = \sqrt{\frac{1}{n-1}\sum_i (dx_i - \mu_{SE,dx})^2}$$

Par exemple, la relation « Superscript » implique deux modèles de différence de tailles (*dh*), Figure 67, (la taille de la base et la taille de l'exposant) et deux autres pour la différence d'alignement (*dy*) (la position de la base et celle de l'exposant).

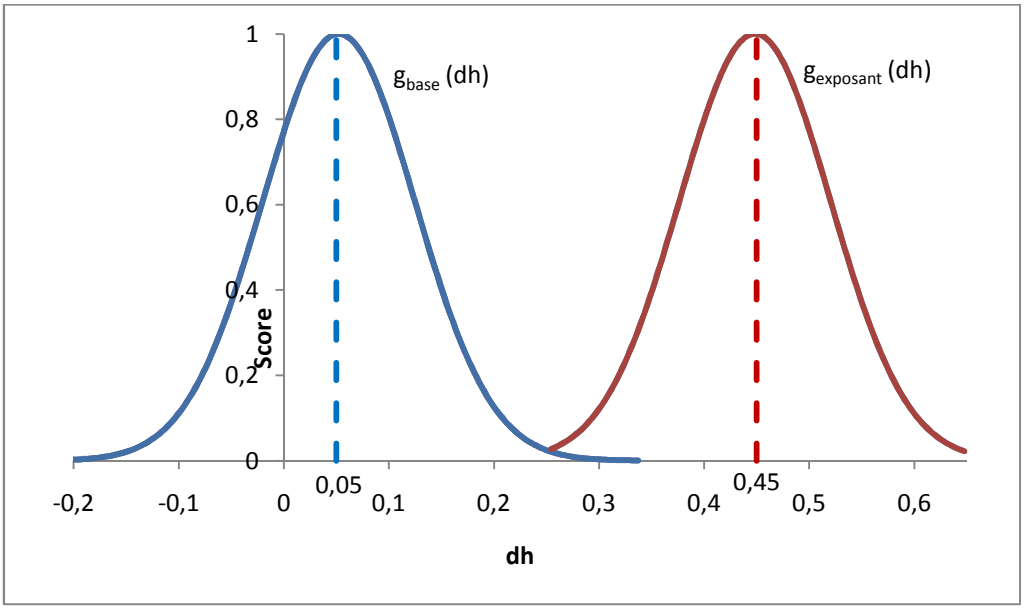



Figure 67 - Modèles gaussiens de la différence taille de la relation « superscript »

Nous pouvons conclure de la Figure 67 que les modèles appris de la relation « Superscript » impliquent que la base ayant une taille proche de celle du parent est très probable $dh \approx 0$. Nous constatons également que l'exposant ayant un *dh* proche de la valeur « 0.45 » est le plus probable. Il est à remarquer

que la taille du parent ($h_{SE}$ calculée en fonction des tailles des fils) n'est pas forcement plus grande que celles des fils, ce qui explique les valeurs négatives de *dh* dans la figure ci-dessus.

#### 4.6.2.2. Estimation du coût structurel probabiliste

Les coûts structurels sont calculés grâce aux probabilités déduites à partir des histogrammes (ou modèles gaussiens). Une sous-expression SE est produite à partir des sous-expressions $\{SE_1,...SE_i,...,SE_N; N = 2,3,4\}$ reliées par la relation R. On note $p(SE|R)$ la probabilité que les sous-expressions formant $SE$ se relient par la relation R, où :

$p(SE|R) = \prod_{i=1}^{N} p(SE_i|R_i^s)$; où $R_i^s$ dénote la sous-relation qui relie le nœud de la relation R et le fils *i*.

Mais en utilisant la règle de Bayes, la probabilité d'une relation qui produit une sous-expression sachant son modèle est donnée par :

$$p(R|SE) = \frac{p(SE|R).p(R)}{p(SE)} \quad (19)$$

Le terme *p(SE)* peut être ignoré car il est constant pour toutes les relations, en utilisant l'équation ci-dessus on obtient la probabilité d'une relation R :

$$p(R|SE) \propto \prod_{i=1}^{N} p(SE_i|R_i^s).p(R) \quad (20)$$

Où p(R) dénote la probabilité à priori de la relation R calculée à partir de la distribution des relations dans la base d'apprentissage.

La probabilité $p(SE_i|R_i^s)$ qu'une sous-expression *SE* soit reliée à la sous expression $SE_i$ produite par la sous relation $R_i^s$ est calculée à partir des modèles *M* appris des sous relations en utilisant ses informations spatiales (*dh, dy*) :

$$p(SE|R^s) = M_{SE,dh}^{R^s}(dh).M_{SE,dy}^{R^s}(dy) \quad (21)$$

Les valeurs $M_{SE,dx}^{R^s}(dx)$ se déduisent soit des histogrammes précédemment calculés, soit des modélisations gaussiennes.

Il est à remarquer que l'application de l'analyse structurelle et syntaxique est simultanée. Les informations spatiales des sous-expressions, ainsi que les coûts structurels sont calculés au fur et à mesure de l'analyse syntaxique.

## 4.7. Analyse syntaxique (Modèle de langage)

Une expression mathématique est produite par un langage bidimensionnel (2D). Malgré l'efficacité des grammaires hors-contexte pour analyser des langages unidimensionnels (1D) (comme les langages de programmation par exemple), l'analyse de tel langage en deux dimensions nécessite des algorithmes et des contraintes pour réduire la complexité et l'ambiguïté [**21**].

Comme les grammaires 2D doivent faire face à des problèmes de performance, le modèle de langage des expressions que nous utilisons est défini par adaptation d'une grammaire 1D. La grammaire utilisée consiste en deux types de règles 1D, une sur l'axe vertical et l'autre sur l'axe horizontal. Les règles verticales (RV) et horizontales (RH) sont appliquées successivement jusqu'à arriver aux symboles élémentaires, ensuite une analyse ascendante est appliquée.

Chacune des règles de production de la grammaire est associée à une relation spatiale qui décrit l'arrangement des éléments intervenant dans la règle. L'application d'une règle est pénalisée selon la probabilité de la relation associée. Donc, chacune des règles de cette grammaire s'active si la relation associée a une probabilité forte par rapport aux autres règles.

Nous avons défini une première grammaire qui ne produit que des expressions de type calculette. Une deuxième grammaire plus large a été définie pour supporter des expressions du corpus RamanReduced, la grammaire complète se trouve dans l'annexe D-2. Bien évidemment, plus la grammaire est stricte vis-à-vis du domaine des expressions, plus les résultats seront précis. A la limite, la grammaire pourrait n'autoriser que les expressions du corpus. Nous allons voir dans l'expérimentation l'effet de la grammaire sur les performances de reconnaissance.

## 4.8. Bloc de décision

Le bloc de décision choisit l'ensemble des hypothèses minimisant le coût global et respectant le modèle de langage. Le coût global est constitué des scores de reconnaissance des symboles et des scores d'activation de toutes les règles qui produisent l'expression finale.

Le coût global d'une expression candidate est celui de la racine de l'arbre relationnel retourné par l'analyseur syntaxique :

$$C_E = C(SE_{racine}) \qquad (22)\ ;$$

où le coût d'un nœud dans l'arbre relationnel est défini par la formule récursive :

$$C(SE_j) = \begin{cases} C_{reco}(hs_j) & \text{; si SE est terminal} \\ \alpha.C_{struct}(R\,|SE_j) + \sum_i C(SE_i) \text{ avec} \bigcup_i SE_i = SE & \text{; sinon} \end{cases} \quad (23)$$

où $C_{struct}(R\,|SE_j) = -log(p(R|SE_j))$ pour un coût probabiliste (cf. §4.6.2), et $C_{struct}(R\,|SE_j) = C_R$ en cas de coût géométrique (cf. §4.6.1). Le coût d'un nœud terminal est le score de reconnaissance de l'hypothèse du symbole : $C_{reco}(hs_j)$.

La nature du facteur α est différente selon le type de coûts structurels utilisés. Les coûts structurels probabilistes sont de même nature que ceux de reconnaissance. Dans ce cas, le facteur α sert à donner plus au moins d'importance aux coûts structurels par rapport aux coûts de reconnaissance.

En ce qui concerne les coûts géométriques, on cherche à ramener le coût de reconnaissance dans le même ordre de grandeur que les erreurs quadratiques en normalisant celles de reconnaissance par la somme des carrés des longueurs des traits *t* de l'hypothèse de symbole ($hs_j$) :

$$Norm(hs_j) = \sum_{i=1}^{n} (longueur^2(t_i)); avec \bigcup_i t_i = hs_j \quad (24)$$

Le facteur α de la formule (23) est représenté par les facteurs $\gamma$ et $\beta$ vus précédemment dans le calcul des coûts géométriques (équation 16). Les formules (15) et (23) deviennent dans le cas des coûts géométriques :

$$C_E = \sum_{i=1}^{n} Norm(hs_i).C_{reco}(hs_i) + \sum_{j=1}^{r} \alpha.C_{struct}(R_j|SE_j) \quad (25)$$

$$C(SE_j) = \begin{cases} Norm(hs_j).C_{reco}(hs_j) & \text{; si SE est terminal} \\ \alpha.C_{struct}(R\,|SE_j) + \sum_i C(SE_i) \text{ avec} \bigcup_i SE_i = SE & \text{; sinon} \end{cases} \quad (26)$$

## 4.9. Conclusion

Nous avons présenté dans ce chapitre l'architecture globale de notre système de reconnaissance d'expressions mathématiques manuscrites en-ligne. Les caractéristiques principales du système proposé résident en deux points :

- L'application simultanée de la reconnaissance, la segmentation, et l'interprétation en transformant le problème de reconnaissance en une minimisation d'une fonction de coût global. Cette fonction est déduite des scores de reconnaissance venant du classifieur des symboles, et des scores structurels venant d'une analyse syntaxique des relations spatiales reliant les symboles de l'expression ;

- L''introduction d'une classe de rejet en proposant un classifieur hybride ou global de symboles. Pour surmonter le problème de l'apprentissage de la classe de rejet, nous avons proposé une méthodologie d'apprentissage global qui permet d'apprendre le classifieur de symboles y compris la classe de rejet. Plusieurs stratégies d'apprentissage global ont été mises au point afin de trouver la plus appropriée au problème de reconnaissance d'expressions mathématiques.

La Figure 68 reprend l'architecture globale du système présentée au début de ce chapitre en résumant les options proposées. Nous pouvons choisir parmi trois types des classifieurs (isolé, hybride ou global) et parmi deux modes de calculs de coûts structurels (géométrique ou gaussien) et finalement le modèle de langage est choisi selon le domaine d'expressions désiré (deux modèles ont été implémentés : calculette et RamanReduced).

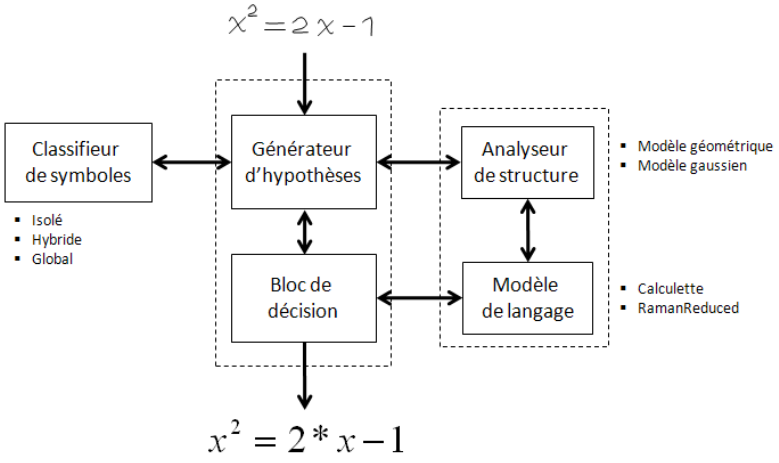



Figure 68 - Architecture globale du système avec les différentes options proposées

Nous allons dans le chapitre suivant présenter les résultats avec les indicateurs de performance proposés dans le chapitre 3 pour de nombreuses configurations de l'architecture proposée. Nous résumons à la fin les performances rencontrées dans la littérature. Enfin, nous conclurons avec un exemple de reconnaissance d'un autre langage 2D (organigrammes) en utilisant le même type d'approche.

# CHAPITRE 5 : EXPÉRIMENTATIONS ET RÉSULTATS

L'objectif des expérimentations réalisées est d'analyser et de mesurer les performances des différentes approches proposées pour la reconnaissance d'expressions mathématiques manuscrites en-ligne. Elles permettront également de régler certains paramètres des différents systèmes étudiés. Après un rappel sur les bases utilisées et sur les mesures d'évaluations, nous décrivons les expérimentations sur :

- La performance des classifieurs isolés.
- L'utilisation d'un modèle de langage pendant l'apprentissage du système.
- L'optimisation des paramètres du reconnaisseur d'expressions.
- Le choix du classifieur de rejet.
- Le choix du modèle structurel.
- L'impact de la base d'apprentissage.
- L'impact de la stratégie de l'apprentissage global.

Finalement, nous présenterons l'application du système sur le problème de reconnaissance d'un autre langage bidimensionnel (les organigrammes de programmation).

## 5.1. L'évaluation

### 5.1.1. Bases de données

Toutes les expérimentations sont effectuées avec les bases des données détaillées dans le chapitre 3. La base CIEL de symboles isolés est découpée en deux parties. La première (180 scripteurs) est utilisée pour l'apprentissage des classifieurs isolés. La deuxième partie (100 scripteurs) sert à évaluer ces classifieurs.

En ce qui concerne les bases d'expressions, la partie utilisée pour l'apprentissage global du système (classifieurs et modèles structurels) est résumée dans le Tableau 15.

Tableau 15 - Constitution des bases d'apprentissage d'expressions

| Base | #Scripteurs | #Symboles | #Expressions |
|---|---|---|---|
| Calculette | 180 | 5448 | 900 |
| RamanReduced_CIEL | 180 | 74160 | 6480 |
| RamanReduced_IROCIEL | 200 | 82400 | 7200 |

Le Tableau 16 résume les bases d'expressions utilisées pour évaluer la performance du reconnaisseur d'expressions mathématiques.

Tableau 16 - Constitution des bases de test d'expressions

| Base | Nature | #Scripteurs | #Symboles | #Expressions |
|---|---|---|---|---|
| Calculette | Synthétique | 100 | 3051 | 500 |
| RamanReduced_CIEL | Synthétique | 100 | 41200 | 3600 |
| RamanReduced_Réelle | Réelle | 10 | 784 | 70 |
| RamanReduced_Wiki_CIEL | Réelle | 166 | 1477 | 211 |

### 5.1.2. Les mesures d'évaluation

Nous nous sommes limités aux trois mesures suivantes pour évaluer les performances du reconnaisseur d'expressions mathématiques (cf. chapitre 3):

- Le taux de segmentation :

$$\text{segRate} = \frac{\text{nombre de symboles bien segmentés}}{\text{nombre total de symboles}}$$

- Le taux de reconnaissance des symboles :

$$\text{recoRate} = \frac{\text{nombre de symboles bien reconnus}}{\text{nombre total de symboles}}$$

- Le taux de reconnaissance d'expressions :

$$\text{expRate} = \frac{\text{nombre d'expressions bien reconnues}}{\text{nombre total d'expressions}}$$

Ces mesures sont utilisées pour évaluer la reconnaissance d'expressions, mais aussi pour valider l'apprentissage.

Il pourrait sembler judicieux de mesurer les performances en termes de rejet des mauvaises hypothèses de segmentation, par exemple grâce à un taux de fausses acceptations ou de vrais rejets. Toutefois, cette évaluation est difficile à cause de la grande quantité et variété des mauvaises hypothèses générées par rapport aux vrais symboles (en moyenne 61 mauvaises segmentations pour chaque symbole d'une expression). Cette mesure n'est pas forcement pertinente dans notre système car celui-ci permet de prendre en compte plusieurs résultats de classification de chaque hypothèse : la bonne réponse (rejet ou symbole) peut être en seconde position et néanmoins retenue par la solution globale.

### 5.1.3. Protocole expérimental

Une expérimentation consiste en deux étapes : l'apprentissage du classifieur puis l'évaluation de la performance du reconnaisseur d'expressions. L'apprentissage peut être isolé ou global.

Dans le cas isolé, le classifieur est entraîné sur la base isolée et ensuite utilisé dans le reconnaisseur d'expressions.

L'apprentissage global est effectué avec une base d'apprentissage d'expressions. Dans un apprentissage libre (sans grammaire) le résultat de la

reconnaissance d'une expression manuscrite est un ensemble de regroupements de traits reconnus comme symboles sans aucune information contextuelle. Dans ce cas, pendant l'apprentissage nous n'utilisons que les taux de segmentation et de reconnaissance des symboles. Ce dernier est le critère que nous avons choisi pour arrêter l'apprentissage. En pratique, le nombre d'itérations est assez large pour assurer la convergence du classifieur. Le classifieur qui correspond à l'itération avec le taux de reconnaissance maximal de la base d'apprentissage est celui conservé pour les tests. Les taux obtenus lors de cet apprentissage ne correspondent pas à la performance finale. En effet le système final utilise une grammaire permettant de n'obtenir que des expressions valides, ce qui n'est pas le cas pendant l'apprentissage. En conséquence, l'évaluation complète d'un tel classifieur se fera en utilisant le classifieur obtenu associé à la grammaire du domaine désiré.

Quant à l'apprentissage contraint (avec grammaire), la grammaire est utilisée au cours de l'apprentissage et les résultats sont des expressions valides conformes au domaine désiré. Dans ce cas, le classifieur conservé en fin d'apprentissage est celui donnant la meilleure performance au niveau du taux d'expressions.

Dans tous les cas, les classifieurs obtenus lors de l'apprentissage sont ensuite utilisés dans le reconnaisseur d'expressions pour être testés sur les bases de test. Il est à noter que les résultats obtenus sur une base de test peuvent se différencier d'une expérimentation à une autre. Cela vient du fait que les expérimentations ont été réalisées tout au long des travaux de thèse, et donc les conditions expérimentales sont différentes selon l'évolution du système. Par exemple, les paramètres $t$ et $s$ (décalage et ratio de taille) vus précédemment dans le chapitre 4 ont changé au fur et à mesure sans vouloir les optimiser explicitement.

### 5.1.4. Classifieurs utilisés

Nous avons testé dans ces expérimentations trois classifieurs différents. Pour le premier, nous avons choisi un perceptron multicouches (PMC, voir description section 4.3.2.1), avec une seule couche cachée utilisant 100 neurones. Le second classifieur est un réseau de neurones à convolution (TDNN, voir description section 4.3.2.5), avec une couche cachée à sept neurones. Chacun de ces neurones voit une fenêtre de vingt points du signal d'entrée avec un décalage de 5 points d'un neurone à un autre. La partie PMC comporte aussi une couche cachée à 100 neurones. Le dernier classifieur utilisé est un séparateur à vaste marge (SVM, voir description section 2.2.4) utilisant un noyau Gaussien (C=100 et SDT=10).

Pour les classifieurs isolés, le nombre de sorties est égale au nombre de classes des symboles. Une sortie additionnelle est ajoutée s'il s'agit d'un classifieur global avec rejet.

En ce qui concerne le classifieur de rejet de l'architecture hybride (voir section 4.3.3.2), nous utilisons soit un PMC avec seulement 50 neurones dans la couche cachée, soit un TDNN pour le cas où le classifieur cible est aussi un TDNN. D'autres types de classifieurs de symboles peuvent être utilisés puisque celui-ci est appris indépendamment du reste du système dans le cadre de cette architecture hybride. Le Tableau 17 compare les résultats obtenus avec ces trois classifieurs.

## 5.2. Performances isolées

L'objectif des expérimentations sur les classifieurs isolés est de :

- Assurer le bon choix de la topologie et des paramètres des classifieurs.
- Obtenir des classifieurs isolés qui vont être utilisés dans le système de reconnaissance d'expressions. Celui-ci pourra servir comme système de référence de base.
- Obtenir un classifieur cible pour faire partie d'un classifieur hybride.

Vu le nombre relativement petit d'exemples de chaque classe de symboles, nous avons considéré la base de test en tant que base de validation. Nous affichons la meilleure performance obtenue sur cette base, en retenant la topologie optimale proposée dans la thèse d'Emilie Poisson [**74**]. Il n'y a donc pas d'adaptation spécifique des méta-paramètres à nos bases et donc pas de risque de biais correspondant. Les résultats sur l'ensemble de 223 symboles et les sous groupes détaillés dans le chapitre 3 sont présentés dans le Tableau 17 (un classifieur différent est appris séparément pour chaque groupe de symboles). Dans ce tableau, la performance d'un classifieur isolé est simplement mesurée par la précision totale : le nombre d'exemples bien classés sur le nombre total d'exemples.

Le Tableau 17 montre un bon potentiel en faveur des classifieurs de type réseaux des neurones. Le taux de reconnaissance est affecté par le nombre des classes et leurs complexités. Par exemple, en ce qui concerne le TDNN il décroît de 96.5% pour le groupe le moins nombreux (les chiffres) à 79.41% pour l'ensemble de tous les symboles.

Il est à remarquer que ces performances sont obtenues individuellement sur chacun des groupes. En effet, dans une application de reconnaissance d'expressions mathématiques, les classes issues de ces différents groups du domaine considéré doivent être présentes simultanément.

Une solution pour améliorer les performances de chaque classifieur serait d'étudier la complémentarité de ceux-ci en mettant en œuvre des techniques de combinaison. Nous avons considéré que cette problématique était en dehors des objectifs de ces travaux.

Tableau 17 - Performances des classifieurs isolés sur différents sous-ensembles de la base CIEL isolée

| Groupe | #Symboles d'apprentissage | #Symboles de test | #Classes | TDNN% | PMC% | SVM% |
|---|---|---|---|---|---|---|
| Tous les symboles | 40128 | 22294 | 223 | 79,41 | 77,3 | 79,42 |
| Alphabet Grec | 5400 | 3000 | 30 | 88,52 | 86,8 | 85,88 |
| Symboles élastiques | 6660 | 3698 | 37 | 91,94 | 91,2 | 89,13 |
| Flèches et opérations binaires | 10619 | 5900 | 59 | 94,27 | 93,9 | 92,51 |
| Fonctions | 6298 | 3500 | 35 | 93,5 | 93 | 87,71 |
| Alphabet latin majuscule | 4680 | 2600 | 26 | 96,4 | 95,3 | 94,85 |
| Alphabet latin miniscule | 4674 | 2600 | 26 | 93,15 | 92,1 | 92 |
| Chiffres | 1800 | 1000 | 10 | 96,5 | 96,1 | 96,6 |

Dans les corpus d'expressions choisies pour ces travaux, nous avons considéré deux sous-ensembles de classes qui correspondent aux corpus Calculette (15 classes) et RamanReduced (34 Classes), cf. chapitre 3.

Nous constatons à la lecture du Tableau 18 que les trois classifieurs ont quasiment les mêmes performances sur la base de test des deux groupes. Cependant, le comportement du classifieur peut être différent dans un contexte de reconnaissance d'expressions. En effet, lors de la reconnaissance d'une expression complète, on ne s'intéresse pas seulement à la classe reconnue mais aussi à son score de reconnaissances (et aux scores des classes suivantes) et surtout au comportement du classifieur face aux hypothèses de symboles invalides (pour la capacité de rejet).

Tableau 18 - Performance des classifieurs isolés sur les groupes "Calculette" et "RamanReduced"

| Classifieur | Calculette_Iso Test % (15 classes) | RamanReduced_Iso Test % (34 classes) |
|---|---|---|
| PMC | 96,6 | 95,4 |
| TDNN | 96,8 | 95,7 |
| SVM | 96,7 | 96,2 |

## 5.3. Système de référence

Nous considérerons comme système de base, celui correspondant à l'architecture complète présentée dans la Figure 52 du chapitre 4, mais avec un classifieur de symboles sans capacité explicite de rejet et entraîné avec une base de symboles isolés. Cela correspond à la plupart des approches proposées

dans les systèmes actuels. Nous avons intégré les classifieurs présentés ci-dessus dans l'architecture globale du système.

Le Tableau 19 montre la performance de ce reconnaisseur d'expressions sur les bases de test. Dans le cas de la base de test Calculette, plus de 96% des symboles sont bien segmentés et plus de 94% sont bien reconnus en utilisant un TDNN comme classifieur de symboles. Toutefois, les expressions de type calculette sont relativement faciles car elles ne comportent pas de structures 2D. De plus, la grammaire utilisée dans ce cas est très stricte (cf. chapitre 4), ainsi les contraintes associées ont permis sur cette base de test de bien reconnaître toutes les occurrences du signe « = ». On observe que 81.1% des expressions sont bien reconnues. Les PMC et SVM réalisent des performances proches.

La difficulté de la disposition 2D des symboles est plus présente avec les autres bases de test, ce qui détériore les performances du système. Malgré la bonne performance des classifieurs en contexte de symboles isolés, le taux de reconnaissance des symboles en contexte d'expressions ne dépasse pas 74% dans le meilleur cas. En effet avec une grammaire plus large, le problème des hypothèses invalides perturbe la reconnaissance des expressions.

La Figure 69 illustre ce problème sur un exemple de reconnaissance de l'expression manuscrite $\frac{a+b}{c+d}$. Les symboles du numérateur (a b +) sont pratiquement très bien alignés dans une relations de type « opérateur ». Le 'a' est reconnu avec un bon score (0.99), mais le classifieur hésite sur la reconnaissance de l'opérateur en donnant un score faible à la classe '+' (0.36). Supposons que le 'b' soit également bien reconnu avec un score de 0.99. On obtient alors un coût de reconnaissance pour ces trois symboles :

$$C_{reco}(a+b) = log(0.99) + log(0.36) + log(0.99) = log(0.35).$$

Par ailleurs, la segmentation qui regroupe les traits du 'a' et du '+' est reconnue en tant que 'cos' avec également un score de 0.99. Ce score élevé est possible car cette donnée se situe dans une partie de l'espace d'entrée pour laquelle aucun exemple en apprentissage n'a été rencontré. De plus, du point de vue structurel les symboles 'cos' et 'b' sont très bien alignés. Cette mauvaise segmentation obtient alors un score de reconnaissance plus élevé (donc un coût plus faible) :

$$C_{reco}(\cos b) = log(0.99) + log(0.99) = log(0.98).$$

Le résultat final serait alors $\frac{\cos b}{c+d}$ à cause du comportement imprévisible de classifieur face aux segmentations invalides.

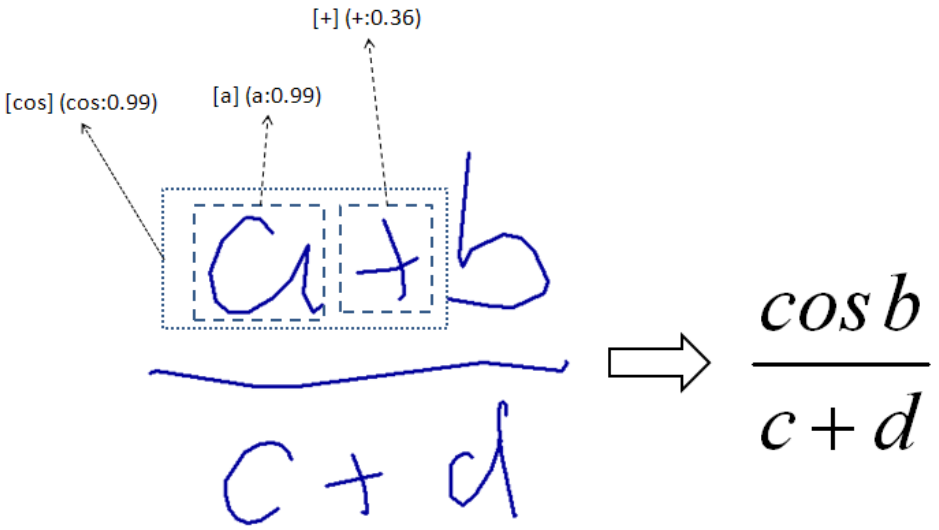



Figure 69 - Exemple du comportement imprévisible face aux mauvaises segmentations

Le Tableau 19 montre qu'un reconnaisseur avec un classifieur SVM a une performance supérieure à celles avec un PMC ou un TDNN surtout au niveau expression (45.% sur la base RamanReduced_CIEL et 28.6 sur la base RamanReduced_réelle). On peut en déduire que le SVM a une capacité de modélisation intrinsèque supérieure au TDNN ou au PMC ce qui lui permet de mieux gérer les situations non apprises et de réaliser une performance supérieure[10]. Malgré la performance relativement bonne d'un système avec un SVM, nous nous intéresserons plus aux TDNN et PMC pour leur propriété intéressante d'apprentissage itératif ce qui facilite leur intégration dans le système avec un schéma d'apprentissage global. De plus, les TDNN et PMC sont nettement meilleurs en termes de mémoire occupée et de temps de traitement.

Nous pouvons conclure de cette expérimentation qu'il est indispensable d'ajouter la capacité de rejet au classifieur de symboles. Ceci est réalisable avec un PMC ou un TDNN grâce à l'apprentissage global. Un reconnaisseur utilisant un SVM n'est pas directement capable d'apprendre le rejet itérativement, sauf si un classifieur spécialisé de rejet (PMC ou TDNN) est couplé avec le SVM dans le cadre d'un schéma de classifieur hybride.

Nous allons dans la suite nous limiter à présenter les résultats obtenus par un reconnaisseur d'expressions utilisant un TDNN comme classifieur de symboles. Sachant que les expérimentations ont montré des comportements et performances très proches avec un PMC ou un TDNN.

[10] Cette propriété est due au noyau Gaussien utilisé qui fait diminuer le score des exemples éloignés des vecteurs supports [**103**]

Tableau 19 - Performance du reconnaisseur de référence sur les bases de test

Calculette

| Base | segRate | recoRate | expRate |
|---|---|---|---|
| TDNN | 96,6 | 94,3 | 81,1 |
| PMC | 95,2 | 93,2 | 79,5 |
| SVM | 88,6 | 87,5 | 82,2 |

RamanReduced_CIEL

| Base | segRate | recoRate | expRate |
|---|---|---|---|
| TDNN | 64,2 | 62,2 | 25,6 |
| PMC | 75 | 73,1 | 31,2 |
| SVM | 74,6 | 73,5 | 45,5 |

RamanReduced_réelle

| Base | segRate | recoRate | expRate |
|---|---|---|---|
| TDNN | 50 | 46,6 | 11,4 |
| PMC | 61 | 56,1 | 12,9 |
| SVM | 67 | 63,7 | 28,6 |

RamanReduced_Wiki_CIEL

| Base | segRate | recoRate | expRate |
|---|---|---|---|
| TDNN | 48,9 | 46,2 | 18 |
| PMC | 48 | 46,3 | 15,6 |
| SVM | 73,5 | 72,7 | 51,6 |

## 5.4. Optimisation des paramètres du reconnaisseur d'expressions mathématiques

Dans cette section nous présentons comment optimiser les différents paramètres communs aux stratégies étudiées ensuite. Ces mêmes optimisations peuvent être conduites quelque soit le classifieur utilisé (avec ou sans rejet) et quelque soit la stratégie d'apprentissage global (section 5.5). Les paramètres concernés sont : le facteur α de pondération des coûts structurels et de reconnaissance (représenté par les paramètres $\beta,\delta$ pour le cas des coûts géométriques), et le nombre de candidats retenus par le classifieur (représenté par le paramètre *topN* et le seuil *k*).

Pour éviter une recherche exhaustive des valeurs optimales des paramètres, nous avons suivi une méthode simple. Au moment d'optimiser un des paramètres nous fixons les valeurs des autres (à leurs valeurs optimales s'ils étaient déjà optimisés).

Il faut souligner que notre système de reconnaissance d'expressions comporte un nombre important de paramètres libres qui peuvent poser problèmes lors du passage à l'échelle de la solution proposée. Toutefois, la valeur de certains de ces paramètres n'a pas été particulièrement optimisée pour les expressions mathématiques (par exemple la topologie du TDNN) mais est commune à plusieurs applicatifs. D'autres sont calibrés en utilisant une base de validation (le facteur α, nombre de candidat du classifieurs, etc.) et

puis testés sur une base indépendante. Dans le cas de notre étude ces méta-paramètres ont été réglés sur les bases synthétiques puis utilisés pour l'évaluation sur les bases réelles.

### 5.4.1. Optimisation des paramètres $\beta,\delta$

Pour cette étude, les paramètres qui déterminent le nombre des candidats retenus par le classifieur sont fixés à : *k=0.9* et *topN=3*. Nous avons choisi au hasard un sous ensemble de la base de test RamanReduced_CIEL de 10 scripteurs (360 expressions) pour optimiser les paramètres $\beta,\delta$ . Ces paramètres participent à équilibrer la différence entre les scores structurels géométriques et les scores de reconnaissance (cf. chapitre 4). Pour trouver la valeur optimale de ces deux paramètres nous avons procédé comme suit : tout d'abord la valeur de $\delta$ est fixée empiriquement à 0.25, ensuite la valeur de $\beta$ varie dans l'intervalle entre 0 et 1. La valeur optimale est celle qui maximise le taux de reconnaissance d'expressions. La courbe sur la Figure 70 montre l'évolution du taux de reconnaissance d'expressions (expRate) en fonction de $\beta$. Le taux d'expressions approche 52% quand $\beta$=0.6.

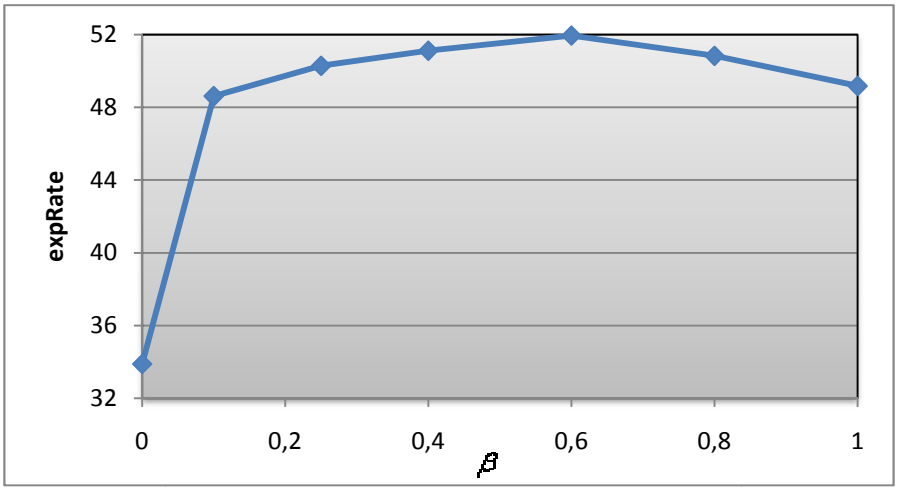



Figure 70 - Evolution de taux de reconnaissance d'expressions en faisant varier le paramètre $\beta$ ($\delta$=0.25 , k=0.9 , topN=3)

Puis, en fixant $\beta$=0.6, la valeur de $\delta$ est déterminée de manière analogue. La courbe de la Figure 71 montre que la valeur optimale du paramètre $\delta$ et de 0.4 avec un taux de reconnaissance d'expressions à 53.6%.

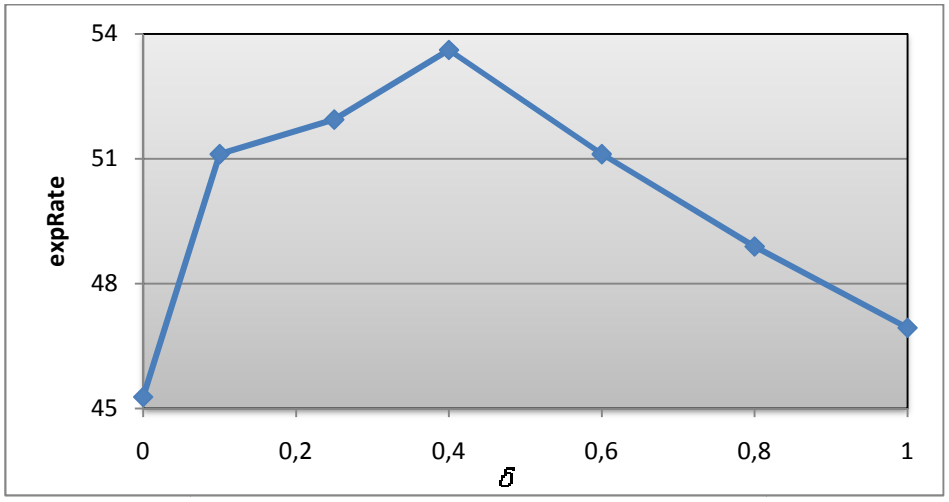



Figure 71 - Evolution de taux de reconnaissance d'expressions en faisant varier le paramètre $\delta$ ( $\beta$=0.6 , k=0.9 , topN=3)

Cette procédure de réglage est bien sûr sous-optimale, son efficacité va dépendre de la complexité du paysage de la fonction d'optimisation. Etant données les formes très simples trouvées sur la Figure 70 et la Figure 71, nous pouvons penser qu'une méthode plus complexe ne modifierait pas beaucoup les résultats. Ces deux paramètres sont optimisés une fois pour toute : dans toutes les expérimentations qui utilisent la modélisation géométrique nous aurons : $\delta$=0.4 et $\beta = 0.6$ .

### 5.4.2. Optimisation du nombre de candidats retenus par le classifieur (topN et k)

Le nombre de candidats considérés en résultat de reconnaissance d'une hypothèse de symbole joue un rôle très important. En effet, les candidats qui ne sont par retenus à cette étape, ne pourront plus être utilisés lors de la recherche de la meilleure solution.

Les classifieurs vont avoir des comportements différents selon la stratégie d'apprentissage. Le classifieur peut être très strict en donnant un score élevé au premier candidat. Dans ce cas le seuil *k* doit être suffisamment grand pour considérer d'autres candidats. Par contre, le classifieur peut être plus flexible en donnant des scores plus étalés sur les premiers candidats. Dans ce cas, un petit seuil va diminuer le nombre de candidats retenus.

La Figure 72 montre l'évolution du taux de reconnaissance d'expressions sur la base de test RamanReduced_CIEL en fonction du nombre moyen de candidats retenus du classifieur (un TDNN avec une sortie pour la classe rejet).

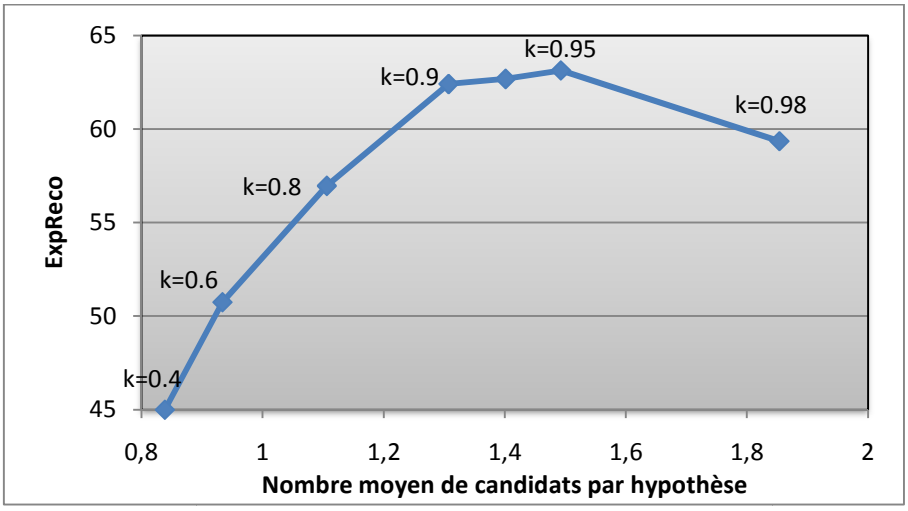



Figure 72 - Evolution de taux de reconnaissance d'expressions en fonction du nombre moyen de candidats conservés lors de la reconnaissance de symboles ( $\beta$=0.6 , $\delta$=0.4 , topN=5)

Cette courbe est obtenue en fixant le paramètre *topN* à cinq candidats au maximum. Ensuite, on fait varier le seuil *k* entre 0 et 1 pour considérer plus ou moins de candidats. Considérer un seul candidat limite la capacité du classifieur en imposant une seule identité à chaque hypothèse. De plus, dans certain cas le classifieur peut répondre « rejet » en première place. Puisque le rejet n'apparaît pas dans les résultats, le nombre moyen de candidats décroit à moins de 1 pour les très petits seuils (k=0.4 et k=0.6 dans la courbe). D'un autre coté, un grand nombre de candidats donne plus de liberté à l'analyseur structurel parmi les candidats qui sont possiblement ambigus. Mais il faut remarquer qu'il ne faut pas que ce nombre soit très grand au risque de manquer des interprétations correctes. L'expérimentation montre que 1.5 candidats en moyenne mène à la meilleure performance. Ce nombre correspond à un seuil cumulé *k=0.95*.

La Figure 73 montre comment le résultat de reconnaissance est affecté par le choix du paramètre *k*. Dans le cas (a) le positionnement du 'd' favorise qu'il soit interprété structurellement comme un indice du symbole précédent. Mais avec un seuil inférieur à 0.96 cette erreur est évitée grâce au bon score de reconnaissance du '+' et au fait que la relation $+_d$ n'est pas autorisée dans la grammaire. Par contre, si la valeur du *k* choisie est très élevée, un deuxième candidat 'x' entre en concurrence avec le '+'. Mais le coût structurel de $x_d$ est plus faible (meilleur) que celui considérant l'hypothèse '+ d'. Il ne faut donc pas que ce seuil soit très grand afin d'éviter ce genre d'instabilité.

Dans le cas où un symbole n'est pas très bien écrit (le '4' dans l'exemple (b)), un petit seuil n'est pas favorable. Par exemple si le seuil est inférieur à 0.88, le seul candidat sera le 'y'. Donc, pour assurer la présence du '4' dans la

liste des candidats il faut un seuil plus grand que 0.88 pour bien reconnaître la sous-expression *4ac*.

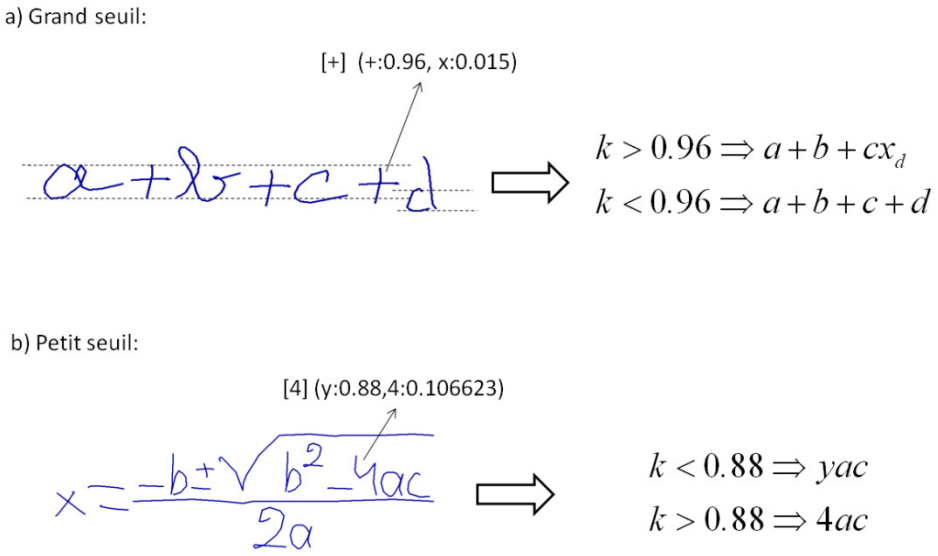



Figure 73 - Effet du choix du paramètre k

Cette optimisation est effectuée pour chaque nouveau classifieur afin d'optimiser au mieux la valeur *k* du seuil.

### 5.4.3. Optimisation du paramètre α

Ce paramètre est utilisé avec les coûts structurels probabilistes. Il sert à pondérer différemment les coûts de reconnaissance et ceux structurels avec une modélisation probabiliste. Contrairement aux paramètres ( $\beta, \delta$ ), le paramètre α est ré-optimisé à chaque changement de modèle structurel ou de classifieur. En effet dans le cas d'une modélisation probabiliste les scores structurels et ceux de reconnaissance ont la même nature (probabiliste) et sont du même ordre de grandeur. Donc, un petit changement du comportement d'un coté affecte énormément le comportement de la fonction de coût global.

## 5.5. Apprentissage global

### 5.5.1. Effet du choix du modèle de grammaire pendant l'apprentissage (modèle libre Vs modèle contraint)

Dans un apprentissage avec un modèle de grammaire, celle-ci est utilisée pour contraindre les résultats de reconnaissance. Par contre, un apprentissage avec un modèle libre n'imposera aucune contrainte aux résultats obtenus. Dans les deux cas, les résultats obtenus sont ensuite comparés de la même façon avec la vérité terrain pour entraîner le classifieur (cf. chapitre 4). Les résultats dans le Tableau 20 sont obtenus en utilisant la modélisation géométrique avec un TDNN comme classifieur ; le modèle contraint utilise la grammaire RamanReduced (cf. Annexe D-2).

Tableau 20 - Performance du reconnaisseur en fonction de modèle de langage utilisé pendant l'apprentissage

RamanReduced_CIEL

| Modèle de langage en apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| Libre | 86,9 | 84,6 | 61,8 |
| Contraint | 82,2 | 81,4 | 47,3 |

RamanReduced_Réelle

| Modèle de langage en apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| Libre | 78,7 | 72,5 | 27,1 |
| Contraint | 70,1 | 63,9 | 22,9 |

Le Tableau 20 montre qu'un apprentissage libre est beaucoup plus bénéfique par rapport à un contraint. Ce résultat peut sembler a priori inattendu. L'explication se situe sûrement dans la diversité des situations rencontrées lors de l'apprentissage non contraint par la grammaire, ce qui permet de bien apprendre la classe de rejets. A l'inverse, lorsque le résultat doit respecter la grammaire, il supporte moins de liberté de segmentation, et la classe rejet sera moins bien apprise.

### 5.5.2. Influence de l'utilisation de scripteurs virtuels

Le choix de la base d'apprentissage est essentiel pour pouvoir bien entraîner le classifieur de symboles. Elle doit être bien représentative du domaine du corpus, mais aussi comporter une bonne variabilité de styles d'écriture. Pour le moment l'apprentissage global s'appuie sur une base d'expressions synthétiques dans le but d'avoir une grande quantité d'expressions. Comme nous l'avons vu dans le chapitre 3, une des limites du générateur d'expression réside dans la répétition du même échantillon pour toutes les occurrences d'un symbole.

Les résultats présentés Tableau 21 sont obtenus en utilisant deux bases d'expressions pour l'apprentissage. La première, RamanReduced_CIEL, avec la contrainte des échantillons répétés. La deuxième, RamanReduced_IROCIEL, est générée avec des scripteurs virtuels depuis les bases isolées IROCIEL. Dans ces expériences, un reconnaisseur avec une modélisation gaussienne pour le modèle structurel est utilisé. Dans les deux cas, les paramètres du reconnaisseur ont été optimisés sur la base de test RamanReduced_CIEL.

Tableau 21 - Performance du reconnaisseur en fonction de la base d'apprentissage utilisée

RamanReduced_CIEL (test)

| Base d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| RamanReduced_CIEL | 91,4 | 88,7 | 64,9 |
| RamanReduced_IROCIEL | 94,3 | 92,1 | 71 |

RamanReduced_réelle

| Base d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| RamanReduced_CIEL | 83,9 | 76,2 | 27,1 |
| RamanReduced_IROCIEL | 86 | 81,4 | 38,6 |

RamanReduced_Wiki_CIEL

| Base d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| RamanReduced_CIEL | 82,5 | 78 | 45,5 |
| RamanReduced_IROCIEL | 89,5 | 86,5 | 52,6 |

En comparant les taux de reconnaissance individuels des lettres et chiffres entre les deux bases d'apprentissage, nous constatons que l'introduction des scripteurs virtuels améliore la reconnaissance de ces classes. Par exemple, le taux de reconnaissance du '1' augmente de 88.6% à 94%, et celui du 'n' de 84% à 97%. Puisque les lettres et les chiffres sont les symboles les plus fréquents dans les expressions, cette amélioration est accompagnée d'une amélioration de performance globale du système. Nous constatons une amélioration de 7% sur la base RamanReduced_CIEL, de 7% RamanReduced_Wiki_CIEL et de 11% sur la base RamanReduced_réelle.

### 5.5.3. Choix de la stratégie d'apprentissage

Tous les systèmes présentés précédemment sont entraînés avec la stratégie d'apprentissage global pur. Pour profiter encore plus de la base de symboles isolés, nous avons cherché à étendre l'apprentissage du classifieur de symboles, soit en poursuivant celui-ci après la phase globale avec un apprentissage isolé (globo-isolé), soit en l'initialisant par la base isolée (iso-global), ou même en considérant les symboles de la base isolée en tant qu'expressions (iso en global) (cf. section 4.4).

#### 5.5.3.1. Apprentissage globo-isolé

Dans cette stratégie, le classifieur appris globalement est renforcé par un apprentissage isolé. L'objectif est d'améliorer la reconnaissance des classes qui sont peu présentes dans la base d'expressions. Nous avons considéré un classifieur global appris sur la base d'apprentissage RamanReduced_CIEL. Pour optimiser la reconnaissance des symboles isolés, le classifieur a été ensuite entraîné sur la base isolée IROCIEL. Comme variante à cette stratégie, nous avons aussi considéré l'ajout à cette base isolée, d'exemples de rejets choisis aléatoirement au cours de l'apprentissage global.

Le Tableau 22 montre la performance du classifieur étendu appris après une seule itération d'apprentissage isolé, et aussi après un apprentissage isolé complet (jusqu'à la convergence du classifieur).

Tableau 22 - Performance du reconnaisseur en utilisant un classifieur globo-isolé comparé à un classifieur global pur

| | RamanReduced_isolé | RamanReduced_réelle | | |
|---|---|---|---|---|
| Base d'apprentissage | Performance isolé | segRate | recoRate | expRate |
| Global (pur) | 93,5 | 87,7 | 80,2 | 28,6 |
| Globo-Isolé (1 itération) [IROCIEL] | 94,3 | 82,5 | 75,3 | 22,9 |
| Globo-Isolé [IROCIEL] | 95,1 | 59,1 | 53,7 | 11 |
| Globo-Isolé (1 itération) [IROCIEL]+rejet | 94 | 79,5 | 71,4 | 20 |
| Globo-Isolé [IROCIEL]+rejet | 94,7 | 64 | 58,2 | 20 |

Nous constatons à la lecture du Tableau 22 que la performance du classifieur s'améliore en isolé. Par contre, quand il est testé sur la base d'expressions RamanReduced_réelle la performance chute rapidement. Le taux de reconnaissance des symboles décroît de 80.2% à 75.3% après une itération et à 53.7% à la fin de l'apprentissage. Cette chute de performance s'explique par le fait que le classifieur oublie la classe rejet puisque il ne rencontre plus d'exemples de cette classe dans la base additionnelle. En effet, les paramètres du classifieur vont être mis à jour de sorte à améliorer la reconnaissance des autres classes sans considérer le rejet. L'ajout de quelques exemples de rejet dans la base isolée permet de limiter cet effet, néanmoins, on note même dans ce cas une baisse des performances, car ces exemples ne sont pas suffisants pour généraliser cette classe.

#### 5.5.3.2. Apprentissage iso-global / iso en global

L'idée de ces deux stratégies est de profiter de la base isolée pour renforcer l'apprentissage global du classifieur. Un apprentissage iso-global consiste à initialiser le classifieur avec un apprentissage en isolé puis à continuer son apprentissage en global. La seconde stratégie utilise la base isolée directement dans l'apprentissage global en tant qu'expressions à un seul symbole. Dans ces expériences, un reconnaisseur avec une modélisation gaussienne pour le modèle structurel est utilisé.

La première ligne du Tableau 23 est la performance du reconnaisseur d'expressions avec un classifieur appris en global (global pur) sur la base d'apprentissage RamanReduced_CIEL. La deuxième est celle d'un classifieur appris en isolé sur la base CIEL et puis en global sur RamanReduced_CIEL. La dernière ligne est le cas où on considère les deux bases (symboles isolés, expressions) dans l'apprentissage global.

Tableau 23 - Performance du reconnaisseur en fonction de la stratégie d'apprentissage

RamanReduced_CIEL

| Stratégie d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| Global Pur | 91,4 | 88,7 | 64,9 |
| Iso-global | 94,2 | 92,5 | 70,6 |
| Iso en global | 93 | 90,8 | 69,6 |

RamanReduced_réelle

| Stratégie d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| Global Pur | 83,9 | 76,2 | 27,1 |
| Iso-global | 87,8 | 81,1 | 31,4 |
| Iso en global | 78,4 | 74,2 | 32,9 |

RamanReduced_Wiki_CIEL

| Stratégie d'apprentissage | segRate | recoRate | expRate |
|---|---|---|---|
| Global Pur | 82,5 | 78 | 45,5 |
| Iso-global | 87,5 | 82,5 | 44,5 |
| Iso en global | 85,7 | 83 | 49,8 |

Les résultats du Tableau 23 montrent l'intérêt de renforcer l'apprentissage global avec la base isolée. En particulier, l'apprentissage iso-global améliore nettement la performance du reconnaisseur sur les trois bases de test.

## 5.6. Choix du classifieur de rejet

Le classifieur intégré dans le reconnaisseur joue un rôle très important car, en plus de reconnaître les symboles, il guide le processus de la segmentation et participe au calcul de la fonction de coût global. Comme nous l'avons vu, un classifieur appris seulement à partir de symboles isolés a des difficultés face aux segmentations invalides (rejet), ce qui montre l'importance de considérer une classe de rejet dans le reconnaisseur.

Dans cette section, nous comparons trois architectures de reconnaisseur, présentées dans la section 4.3.3 :

1. système isolé : le classifieur de référence appris seulement à partir de symboles isolés.
2. système global : un classifieur avec une classe de rejet appris globalement à partir d'expressions complètes.
3. système hybride : le classifieur de référence (appris en isolé) combiné avec un classifieur de rejet (à deux classes, apprises globalement à partir d'expressions complètes).

Pour mesurer l'apport de l'utilisation du rejet, nous avons entraîné deux classifieurs dans les mêmes conditions (paramétrage, base d'apprentissage, …). Pour le système global un TDNN avec une sortie explicite de rejet a été utilisée. En ce qui concerne l'architecture hybride, deux solutions sont comparées : soit un TDDN comme classifieur cible et un TDNN comme

classifieur de rejet ; soit un SVM comme classifieur cible et un PMC comme classifieur de rejet.

Le Tableau 24 montre la performance du reconnaisseur en utilisant un classifieur global (TDNN, cas b) ) et deux hybrides (TDNN /TDNN et SVM/PMC, cas c) ), comparés aux résultats de référence obtenus précédemment avec les classifieurs appris isolément sans capacité de rejet (TDNN et SVM, cas a) ).

Tableau 24 - Performance du reconnaisseur avec ou sans capacité de rejet

Calculette

| Classifieur de symboles | | SegRate | RecoRate | ExpRate |
|---|---|---|---|---|
| TDNN | a) Isolé | 96,6 | 94,3 | 81,1 |
| | b) Global | 99,2 | 96,2 | 84,2 |
| | c) Hybride | 99,2 | 97 | 87,3 |
| SVM | a) Isolé | 88,6 | 87,5 | 82,2 |
| | c) Hybride | 99,6 | 97,6 | 89,2 |

RamanReduced_CIEL

| Expressions synthétiques | | SegRate% | RecoRate% | ExpRate% |
|---|---|---|---|---|
| TDNN | a) Isolé | 64,2 | 62,9 | 25,6 |
| | b) Global | 86,9 | 84,6 | 61,8 |
| | c) Hybride | 89,8 | 87,4 | 51,8 |
| SVM | a) Isolé | 74,6 | 73,5 | 45,5 |
| | c) Hybride | 73,6 | 72,9 | 50,6 |

RamanReduced_réelle

| Expressions synthétiques | | SegRate% | RecoRate% | ExpRate% |
|---|---|---|---|---|
| TDNN | a) Isolé | 50 | 46,6 | 11,4 |
| | b) Global | 78,7 | 72,5 | 27,1 |
| | c) Hybride | 79,4 | 74,6 | 30 |
| SVM | a) Isolé | 67 | 63,7 | 28,6 |
| | c) Hybride | 67,6 | 64 | 27,1 |

Nous pouvons d'abord vérifier que les résultats de segmentation (SegRate) dépendent bien du reconnaisseur utilisé car la segmentation n'est pas faite en amont indépendamment du reconnaisseur, mais elle est bien couplée à celui-ci.

Ensuite, nous pouvons constater que l'utilisation du rejet apporte un progrès significatif au niveau taux de reconnaissance d'expressions par rapport au système de référence. En effet, l'utilisation du rejet avec le classifieur TDNN permet une amélioration du score de 81.1% à 87.3% sur la base de test Calculette, de 25.6% à 61.8% sur la base de test RamanReduced_CIEL, et de 11.4% à 30% sur la base RamanReduced_réelle.

A l'inverse, quand on compare le SVM isolé et son intégration dans l'hybride, le comportement est différent entre les bases synthétiques et la base

réelle. Il y a une amélioration sur la base synthétique (82.2% à 89.2% pour la Calculette et 45.5% à 50.6% pour RamanReduced_CIEL), et une baisse de performances (28.6% à 27.1%) sur la base RamanReduced_réelle. Nous pensons qu'il conviendrait d'avoir une approche plus rigoureuse pour l'application de la fonction Softmax aux sorties du SVM afin de pouvoir effectivement interpréter les sorties en tant que probabilités. Au final, avec un SVM, l'apport de la classe supplémentaire dans le système hybride (c) n'apporte pas grand chose, et même détériore légèrement les performances sur la base des expressions réelles. On peut en déduire que le SVM utilise sa capacité de modélisation intrinsèque supérieure au TDNN, ce qui lui permet de mieux gérer les situations non apprises (rejet) sans les avoir explicitement apprises.

Par contre l'utilisation du rejet dans le TDNN permet de dépasser les performances jusqu'ici supérieures des architectures utilisant un SVM.

A ce stade, on peut se poser la question de savoir quel classifieur est le meilleur pour faire face au problème de rejet ? En considérant le TDNN et en comparant le système hybride c) avec le classifieur global b), nous constatons que le classifieur global se comporte un peu mieux sur la base RamanReduced_CIEL (61.8%) que le système hybride (51.8%). Par contre, le classifieur hybride est légèrement mieux sur la base RamanReduced_réelle (30% contre 27.1%). De même, l'hybridation est nettement bénéfique sur la base Calculette. On peut avancer l'hypothèse que la bonne performance sur la base Calculette vient du fait que la plupart des symboles se composent d'un seul trait (1.5 traits en moyenne par symbole). Donc en quelque sorte le classifieur de rejet fait le choix assez facilement entre des hypothèses de symboles avec peu de traits en tant que symbole valide, et celles avec plusieurs traits en tant que rejet.

Malheureusement, cela ne sera pas le cas dans une application plus complexe où de nombreux de symboles se composent normalement en plusieurs traits. Un second point intéressant est que la méthode actuellement utilisée pour la combinaison des scores de reconnaissance dans le classifieur hybride permet de contrebalancer une décision très marquée du classifieur de symboles. Par exemple, même quand le classifieur de rejet hésite entre rejet ou non-rejet avec des scores respectifs de 0.55 et 0.45, et que dans le même temps le classifieur cible donne un score de 0.99 pour associer l'hypothèse à une certaine classe, son score restera inférieur et l'hypothèse sera rejetée.

Pour résumer, on peut dire que l'ajout d'un classifieur spécifique pour modéliser les mauvaises segmentations n'est pas une solution toujours bénéfique. Cela dépend des propriétés de modélisation intrinsèque du classifieur et du domaine des expressions. En conséquence et malgré le bon potentiel de l'utilisation d'un classifieur hybride, nous avons choisi d'adopter le classifieur global pour la suite de nos expérimentations.

## 5.7. Coût structurel Vs coût probabiliste

Nous comparons dans cette section les performances du reconnaisseur en utilisant des modélisations géométrique ou probabiliste. Dans les différentes expérimentations, les modèles probabilistes (histogrammes, gaussiens) ont été appris sur les bases synthétiques RamanReduced_CIEL et RamanReduced_IROCIEL.

### 5.7.1. Mesurer la performance de la modélisation structurelle

L'objectif de cette expérimentation est de mesurer la performance du reconnaisseur en ignorant le classifieur de symbole pour comparer l'efficacité des modélisations structurelles utilisées. En s'appuyant sur la vérité terrain des traits, nous attribuons à toutes les hypothèses non valides un coût de reconnaissance maximal. En même temps, un coût nul est associé aux bonnes hypothèses. Ainsi, seuls les coûts relationnels interviennent dans la reconnaissance.

Le Tableau 25 montre les résultats de la reconnaissance en utilisant les trois modèles structurels : géométrique, histogrammes, gaussien. Il est à remarquer que le 100% de taux de reconnaissance des symboles n'est pas atteint à cause des contraintes de génération d'hypothèses. Par exemple, il suffit qu'un symbole ait plus de traits que le nombre maximal autorisé pour qu'il n'apparaisse pas dans les résultats.

Tableau 25 - Performances du reconnaisseur en forçant la bonne reconnaissance des symboles sur la base de test RamanReduced_CIEL

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Géométrique | 99,7 | 99,7 | 93,3 |
| Histogrammes | 99,7 | 99,7 | 94,8 |
| Gaussienne | 99,96 | 99,96 | 96,5 |

Nous pouvons constater à partir du Tableau 25 que la modélisation probabiliste est légèrement meilleure que celle géométrique. Mais dans tous les cas, le système arrive assez facilement à reconnaître la plupart des expressions avec un classifieur idéal de symboles. En l'absence de celui-ci, comme nous l'avons vu dans les autres expériences, les résultats de classification des hypothèses influencent fortement la performance globale du système.

L'utilisation des histogrammes s'est montrée performante. Toutefois, l'utilisation des histogrammes dans la méthode proposée alourdie le système. En effet, il faut 4 histogrammes pour chaque relation avec pour chacun une résolution de 50 valeurs, augmentant donc le nombre de paramètres libres du système. En se contentant dans le modèle géométrique de valeurs prédéfinies pour fixer les positions et tailles des symboles dans les relations, seuls 4 ou 6 paramètres par relation sont nécessaires. C'est aussi le cas pour les modèles

gaussiens où 4 à 6 valeurs seulement sont nécessaires pour modéliser une relation. En conséquence, nous nous limiterons à l'utilisation des modèles gaussiens pour la modélisation probabiliste.

La modélisation gaussienne est surtout bénéfique dans des situations ambigües. Comme le montre l'exemple de la Figure 74 les coûts géométriques sont très sensibles aux variations autour des situations idéales. La sous-expression ' $ab$ ' est reconnue en tant que $a_b$ car le 'b' se décale du positionnement idéal de la relation « hpair » donnant un coût de 95. Il s'approche plus de la situation idéale de la relation indice « subscript » avec un coût relativement faible de 39. De plus, le coût géométrique explose rapidement puisqu'il est calculé à partir des erreurs quadratiques de positionnement et de taille idéaux. Ce problème peut être compensé en réglant les valeurs de décalage et ratio de taille de la relation. Néanmoins, un tel réglage nécessite une investigation exhaustive de la base d'apprentissage.

Par contre, le coût probabiliste est directement déduit de la distribution probabiliste, ce qui permet de prendre en compte les variations de positionnement selon les situations apprises de la base d'apprentissage. Dans cet exemple, la sous-expression ' $ab$ ' est bien reconnue puisque la relation la plus probable sera la relation « hpair ».

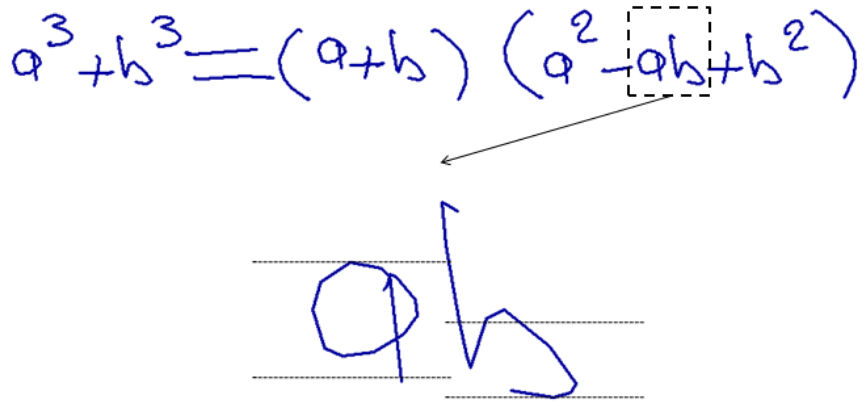



Figure 74 - Les coûts structurels (géométriques et gaussiens) de la sous expression ' $ab$'

### 5.7.2. Résultats

Il est plus approprié de comparer la performance des modèles structurels dans le contexte global du reconnaisseur. Les résultats ci-après sont obtenus en utilisant un classifieur global TDNN entraîné sur la base d'apprentissage

RamanReduced_CIEL. Le Tableau 26 montre les taux de reconnaissance sur les bases de tests après avoir optimisé les paramètres du reconnaisseur.

Tableau 26 - Performances du reconnaisseur avec les modélisations structurelles Géométrique ou Gaussienne

RamanReduced_CIEL

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Géométrique | 92,8 | 90,3 | 64,2 |
| Gaussienne | 91,4 | 88,7 | 64,9 |

RamanReduced_réelle

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Géométrique | 87,7 | 80,2 | 28,6 |
| Gaussienne | 83,9 | 76,2 | 27,1 |

RamanReduced_Wiki_CIEL

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Géométrique | 84,6 | 80,2 | 46 |
| Gaussienne | 82,5 | 78 | 45,5 |

Le Tableau 26 montre que l'apprentissage des relations spatiales améliore légèrement la performance du système sur la base de test synthétisée RamanReduced_CIEL. La bonne performance sur la base synthétisée ne se généralise pas sur les bases réelles. Cette chute de performance vient principalement de la forte dépendance avec les modèles appris sur la base d'apprentissage (la base synthétique) qui ne correspond pas complètement aux situations rencontrées sur la base réelle. Une première solution serait d'apprendre ces relations sur une base réelle importante. Une autre solution pour améliorer les modèles appris serait d'apprendre itérativement les relations spatiales d'une façon discriminante. Cet apprentissage itératif améliorerait à la fois la modélisation des relations de la base d'apprentissage, et augmenterait la capacité de généraliser. Pour ces deux approches, il faudrait disposer d'une grosse base d'expressions réelles afin de pouvoir correctement apprendre ces relations spatiales. Malheureusement nous n'avons pas pu explorer complètement cette voie.

Des expérimentations préliminaires ont néanmoins été effectuées en utilisant l'union des bases RamanReduced_réelle et RamanReduced_Wiki_CIEL pour apprendre les modèles gaussiens. Le Tableau 27 montre une amélioration de performances sur les bases réelles en utilisant un modèle structurel appris sur ces mêmes bases. Cette amélioration est attendue car la même base est utilisée pour l'apprentissage (du modèle relationnel seulement) et pour le test. Néanmoins, tester ces modèles sur la base RamanReduced_CIEL montre que même avec une petite base réelle (281 expressions), nous arrivons à faire aussi bien que le modèle appris sur une grosse base (6480 expressions). Ces résultats montrent l'intérêt d'utiliser de vraies données pour l'apprentissage du système (notamment les modèles

structurels), soit pour apprendre les relations spatiales et bien sûr pour entraîner le classifieur de symboles.

Tableau 27 - Performance du reconnaisseur avec une modélisation structurelle Gaussienne apprise soit à partir de données synthétiques, soit à partir de données rélles

RamanReduced_CIEL

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Gaussienne_synthétique | 91,4 | 88,7 | 64,9 |
| Gaussienne_réelle | 91,6 | 88,5 | 64 |

RamanReduced_réelle

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Gaussienne_synthétique | 83,9 | 76,2 | 27,1 |
| Gaussienne_réelle | 84,6 | 76,5 | 31,4 |

RamanReduced_Wiki_CIEL

| Modélisation structurelle | segRate | recoRate | expRate |
|---|---|---|---|
| Gaussienne_synthétique | 82,5 | 78 | 45,5 |
| Gaussienne_réelle | 83,2 | 78,5 | 48,3 |

## 5.8. Au delà de la reconnaissance d'EM : la reconnaissance d'organigrammes manuscrits en-ligne

Nous allons introduire le problème de reconnaissance d'organigrammes manuscrits en-ligne comme un autre exemple de reconnaissance de structures 2D. Nous nous sommes d'abord intéressés à la collecte d'une base d'organigrammes, étant donné dans ce domaine également, l'indisponibilité de telles données. Quant au système de reconnaissance, nous avons simplement adapté du système précédent la méthode d'apprentissage et les mesures d'évaluation pour reconnaître des organigrammes.

### 5.8.1. Acquisition des organigrammes

Nous avons choisi sept organigrammes de programmation avec différentes complexités. Quelques un décrivent des algorithmes (calcul de factorielle, tri à bulles, apprentissage d'un réseau de neurones). D'autres sont plus simples décrivant des opérations basiques (le carré d'un nombre, somme de N nombres, …). Une liste complète des organigrammes se trouve dans l'annexe B-7. Ces organigrammes contiennent six symboles graphiques, montrés dans la Figure 75. Ils représentent les opérations simples nécessaires pour ce type d'organigramme. De plus, du texte explicatif est fortement présent dans toutes les parties d'un organigramme. Un texte peut être un mot isolé, une phrase simple, une équation mathématique, ou encore un bloc de texte.

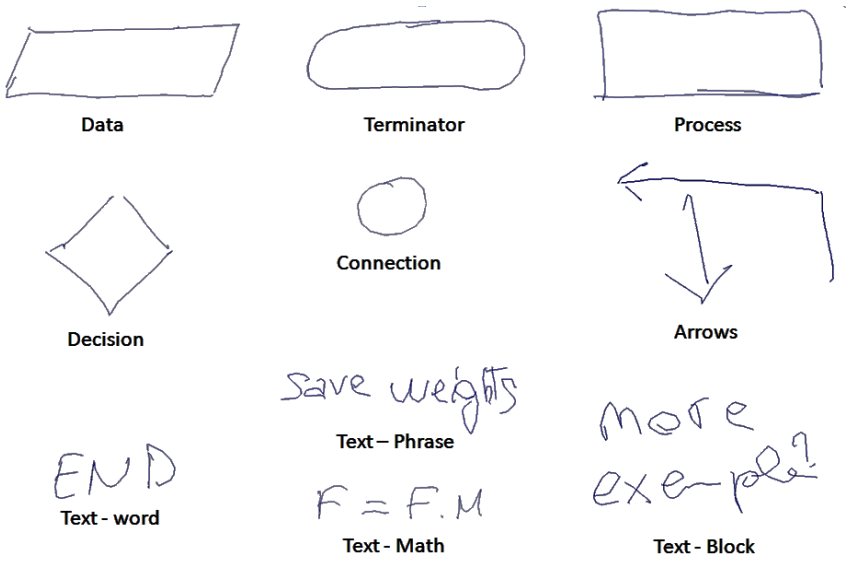



Figure 75 - Exemple des symboles considérés pour la reconnaissance des organigrammes

Des membres de notre équipe de recherche et des étudiants de l'université de Nantes (département de génie électrique et informatique industrielle) ont participé à la saisie des organigrammes en utilisant la technologie Anoto. La Figure 76 montre un exemple d'organigramme et sa version manuscrite. Les scripteurs ont eu pour consigne de copier l'organigramme avec une certaine liberté. Autrement dit, il n'est pas obligatoire de respecter la disposition spatiale des symboles, mais tous les symboles et leurs liens doivent être respectés. De même, il n'y a pas de contraintes d'ordre des symboles ni de direction.

De plus, en observant les organigrammes récoltés nous avons remarqué une grande variété de style de dessin. Quelques scripteurs tendent à dessiner les opérations (processus, décision, …) avec un seul trait. D'autres dessinent les mêmes symboles avec trois traits ou même plus. Il est noté également que quelques scripteurs ne complètent les flèches qu'à la fin de la saisie.

Cette grande liberté et cette variété de dessins d'organigrammes en plus du texte augmente significativement le challenge de reconnaissance de ce type de graphique 2D. A ce stade nous nous sommes focalisés sur le problème de séparation du texte du reste des symboles graphiques de l'organigramme. Nous supposons qu'une fois le texte séparé, il peut être reconnu par une variété de solution classique disponible. La reconnaissance des symboles graphiques est également un problème classique de reconnaissance des formes. Sa complexité dépend du degré de liberté donné aux scripteurs pour dessiner les symboles graphiques.

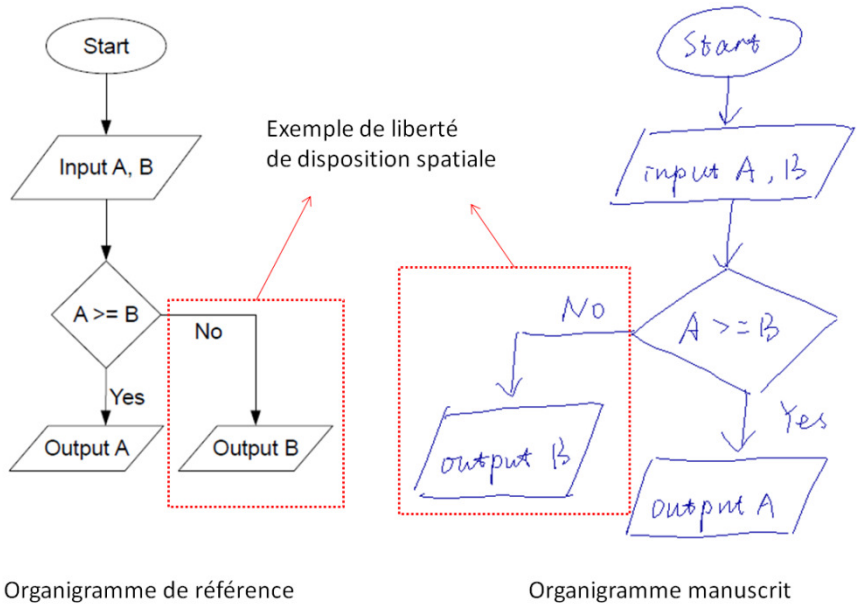



Figure 76 - Exemple d'un organigramme de référence et de sa version manuscrite

### 5.8.2. Une base d'organigrammes manuscrits en-ligne

Un ensemble de 78 organigrammes ont été récoltés. Le tableau montre la constitution des bases d'apprentissage et du test, où 10 scripteurs ont participé à la base d'apprentissage et trois autres à la base de test.

Tableau 28 - Constitution des bases d'organigrammes manuscrits en-ligne

| | # Scripteurs | # Organigrammes | #Symboles |
|---|---|---|---|
| Apprentissage | 10 | 60 | 1287 |
| Test | 3 | 18 | 409 |

Actuellement la sémantique de l'organigramme n'est pas considérée dans la reconnaissance. De plus, il n'existe pas de standard de description des organigrammes qui peut servir en tant que vérité terrain sémantique. En conséquence, les organigrammes ont été étiquetés au niveau symboles et traits. Tous les types de texte sont étiquetés dans la même la classe (***texte***), ce qui est suffisant pour l'évaluation du problème de séparation texte/symboles. De même, toutes les flèches (vers le haut, le bas, la droite, coudées …) sont étiquetées avec la même classe (***flèche***). Le Tableau 29 montre la distribution des symboles dans la base et le nombre des traits de chacune des classes.

Tableau 29 - Nombres et pourcentages des symboles et traits dans les bases d'organigrammes.

| | Apprentissage | | | | Test | | | |
|---|---|---|---|---|---|---|---|---|
| | Symboles | | Traits | | Symboles | | Traits | |
| Processus | 160 | 12,4% | 480 | 8,9 | 51 | 12,5% | 167 | 8,9% |
| Terminateur | 67 | 5,2% | 90 | 1,7% | 23 | 5,6% | 23 | 1,2% |
| Flèche | 420 | 32,6% | 907 | 16,7% | 131 | 32% | 329 | 17,6% |
| Connexion | 39 | 3% | 42 | 0,8% | 10 | 2,4% | 13 | 0,7% |
| Donnée | 72 | 5,6% | 270 | 5% | 23 | 5,6% | 90 | 4,8% |
| Décision | 75 | 5,8% | 291 | 5,4% | 24 | 5,9% | 118 | 6,3% |
| Texte | 454 | 35,3% | 3342 | 61,6% | 147 | 35,9% | 1127 | 60,4% |
| Total | 1287 | 100% | 5422 | 100% | 409 | 100% | 1867 | 100% |

On constate dans ce tableau que le texte représente plus de 35% des composants d'un organigramme. De plus, les traits de ces textes font plus de 60% du nombre total de traits. En conséquence, une méthode classique de segmentation et reconnaissance risque de ne pas être capable de résoudre ce type de problème. De plus, au vu de la complexité de la classe Texte (plus de 7 traits par symbole), nous utiliserons dans certaines expérimentations, en plus du taux de reconnaissance/segmentation des symboles, une évaluation au niveau des traits (pourcentage de traits associés à la bonne classe).

Nous avons proposé deux scénarios pour la reconnaissance d'organigrammes. Dans le premier scénario, nous simplifions le problème en considérant que l'étape de segmentation des symboles est bien effectuée. Donc, le problème est transformé en un problème de reconnaissance de symboles isolés. Cependant, même dans un contexte isolé, nous gardons la définition du problème en tant qu'un problème de séparation du texte des symboles graphique. La base des symboles isolés a été extraite directement des organigrammes récoltés, voir le Tableau 29. Dans le second scénario plus réaliste, nous proposons d'adopter notre reconnaisseur d'expressions pour résoudre le problème complet (segmentation et reconnaissance).

### 5.8.3. Reconnaissance des symboles isolés

Nous avons utilisé les mêmes classifieurs proposés pour la reconnaissance des symboles mathématiques avec la même structure et les mêmes paramètres. Les classifieurs (un TDNN et un SVM) ont été entraînés sur la base de symboles isolés, cf. Tableau 29. Afin de les adapter aux spécificités des organigrammes, une étape supplémentaire de prétraitement est nécessaire.

Tout d'abord le signal d'entrée est normalisé et ré-échantillonné en 30 points (au lieu de 50 pour les symboles mathématiques) résultant en un vecteur de 210 caractéristiques (7 par point) en entrée du classifieur. Ces caractéristiques, identiques aux expressions mathématiques (détaillées dans section. 4.3.1) sont sensibles à l'ordre dans lequel les traits sont saisis. Cette

propriété n'a pas un grand effet sur la reconnaissance des caractères ou du texte car les règles de l'écriture apprises à l'école imposent une certaine stabilité. Cependant, ce n'est pas le cas des symboles graphiques. Comme le montre la Figure 77, l'ordre et la direction des traits varient d'un scripteur à un autre même pour le même symbole selon son habitude d'écriture et le contexte où le symbole se trouve.

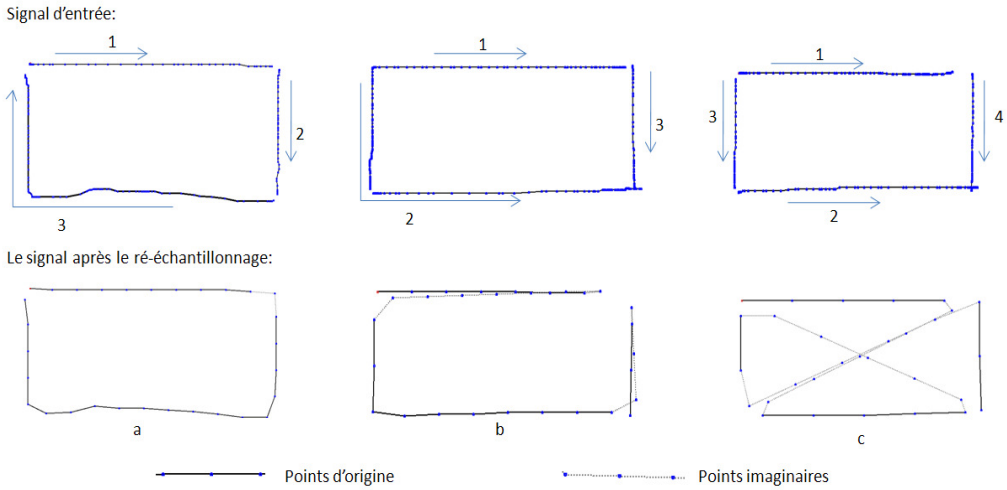



Figure 77 - Exemples de désordre causé par le ré-échantillonnage

Pour éviter cette instabilité une étape additionnelle est appliquée. Elle consiste à réordonner les traits du signal d'entrée en changeant leur ordre par rapport au premier trait de manière à minimiser le nombre de points imaginaires ajoutés. Par exemple, en appliquant cette étape les cas b et c de la Figure 77 vont avoir la même forme que le cas a.

Nous avons d'abord entraîné les classifieurs à reconnaître les 7 classes (texte + 6 symboles). Le Tableau 30 montre que dans un contexte isolé, le SVM réussi à identifier plus de 99% du texte. Les deux classifieurs séparent plus de 95% des textes et des symboles graphiques. Parmi les six symboles graphiques les deux classifieurs en classent correctement 81.3%. La plupart des erreurs viennent de la forte similarité entre les symboles « terminateur » et « connexion ».

Tableau 30 - Performance des classifieurs isolés sur le problème de séparation de texte des symboles graphiques

| | Texte vs. non-texte (2 classes) | | |
|---|---|---|---|
| Classifieur | Texte | Symboles (non-texte) | Symboles graphique (6 Classes) |
| TDNN | 141/147=95,92% | 253/262=96,57% | 81,3% |
| SVM | 146/147=99,32% | 249/262=95,04% | 81,3% |

D'autre part, nous avons entraîné ces classifieurs (TDNN-SVM) uniquement sur les 6 symboles graphiques. Nous pourrions donc imaginer un classifieur à deux étages, où les symboles graphiques sont séparés du texte grâce à un premier classifieur et reconnus par un second. Les résultats sont présentés dans le Tableau 31. Nous remarquons une légère amélioration des taux de reconnaissance des symboles graphiques. Les deux classifieurs (SVM, TDNN) achèvent un taux de 84.4% et 87% respectivement de symboles bien reconnus.

Tableau 31 - Performance des classifieurs isolés sur le problème de reconnaissance des symboles graphiques

| Classifieur | Symboles graphiques(6 classes) |
|---|---|
| TDNN | 84,4% |
| SVM | 87% |

### 5.8.4. Approche globale d'apprentissage et de reconnaissance

Nous avons conçu un système de reconnaissance d'organigrammes basé sur l'architecture globale proposée dans ce manuscrit pour la reconnaissance d'expressions mathématiques. Actuellement le modèle structurel est ignoré dans l'architecture globale (nous n'avons pas développé pour l'instant de grammaire 2D décrivant ce domaine). Donc l'apprentissage ainsi que la reconnaissance se fait sans grammaire (modèle libre). La spécificité des organigrammes est la présence de beaucoup de texte de longueurs différentes. Cette taille peut dépasser le seuil de nombre maximal de traits acceptés par le générateur d'hypothèses. Dans ce cas, le générateur ne pourrait donc pas proposer la bonne segmentation. Pour éviter cet inconvénient, les sous parties de textes sont considérées comme étant une bonne segmentation du texte. Ces sous parties sont obtenus grâce à la sur-segmentation de cette classe. Les parties de textes mal étiquetées sont utilisées pour l'apprentissage global du système. Les morceaux de texte qui apparaissent dans une mauvaise segmentation sont également appris.

Cette architecture est donc moins puissante pour la reconnaissance d'organigrammes en comparaison de sa capacité à reconnaître des expressions mathématiques à cause de l'absence de modèle syntaxique. Néanmoins, l'expérimentation nous a montré un bon potentiel, non seulement pour le problème de séparation du texte, mais aussi pour la segmentation des organigrammes.

Le système a été entraîné sur la base d'apprentissage des organigrammes et testé sur celle de test (cf. le Tableau 28). Puisque la modélisation structurelle n'est pas présente, les résultats de reconnaissance ont été évalués au niveau des traits. La performance est alors mesurée par la capacité du système à identifier si un trait appartient au texte ou aux symboles graphiques.

Pour affiner l'analyse des résultats, nous séparons cette mesure en deux mesures d'évaluation. La première considère les traits des segmentations correctes, nous obtenons le taux de traits totalement identifiés. Pour la seconde mesure, le taux des traits partiellement identifiés même si le symbole auquel ils appartiennent n'est pas correctement segmenté. Dans tous les cas, les traits du texte sont considérés bien segmentés même si ils ne présentent qu'une sous partie d'un bloc de texte, (car il est possible de ne pas pouvoir bien segmenter en un seul bloc de texte à cause de la limite de nombre maximal de traits par symbole). Ainsi, prenons l'exemple d'un symbole composé de 4 traits. Si ces quatre traits et eux seuls sont bien associés pour reconnaître le bon symbole, alors ils participent à la comptabilisation dans la rubrique « Totalement », si par contre seuls 3 des traits ont été associés pour former le symbole, alors ils participeront à comptabilisation dans la rubrique « Partiellement ».

Tableau 32 - Taux de reconnaissance des traits de la base de test d'organigrammes

| Totalement | | Partiellement | |
|---|---|---|---|
| Traits du texte | Traits des symboles | Traits du texte | Traits des symboles |
| 83,3% | 41,22% | 89,44% | 56,62% |

Le Tableau 32 montre que l'architecture globale réussit à correctement segmenter 83.3% des traits provenant des textes d'organigrammes. En ce qui concerne les traits issus de symboles, 41.22% sont bien segmentés et reconnus, cela représente 45.8% des symboles graphiques bien segmentés, parmi lesquels 80% sont bien reconnus. C'est principalement à cause de la forte similarité entre quelques symboles graphiques (flèche horizontale) d'une part et des sous parties de symboles d'autre part (partie haute horizontale d'un processus).

Les expérimentations nous ont montré que l'architecture globale proposée peut être assez facilement adaptée à la reconnaissance de langages 2D très différents des expressions mathématiques. Néanmoins, il est clair que la modélisation structurelle est indispensable pour éviter nombre d'ambiguïtés et améliorer la performance du système global. En effet, l'analyse contextuelle appliquée dans notre système est bien adaptée aux structures 2D de nature récursive (expressions mathématiques, équation chimiques, caractères chinois, etc.) qui peuvent être facilement décrites par une grammaire hors contexte et représentées par des arbres. Nous pensons que l'utilisation d'une grammaire de graphe pourrait être plus efficace pour étendre la capacité du système à reconnaître des structures 2D non-récursives décrites par un graphe (organigrammes, schémas électriques, etc.). L'arbre des expressions mathématiques en serait alors un cas particulier.

## 5.9. Conclusion

Nous avons dans ce chapitre mis en œuvre les différentes approches et stratégies proposées pour la reconnaissance d'expressions mathématiques manuscrites en-ligne. Les résultats obtenus des expérimentations réalisées ont montré l'importance de la présence d'une classe de rejet dans le classifieur de symboles implémenté dans l'architecture globale du système.

La Figure 78 montre l'évolution de la performance du système en appliquant les différentes approches proposées. Les courbes représentent la moyenne pondérée obtenue sur les trois bases de test. Le reconnaisseur optimal est obtenu grâce à la base d'apprentissage RamanReduced_IROCIEL (scripteurs virtuels) qui a participé à entraîner le classifieur global dans un schéma d'apprentissage global classique. La modélisation structurelle est probabiliste, elle est obtenue en modélisant les relations spatiales avec cette même base.

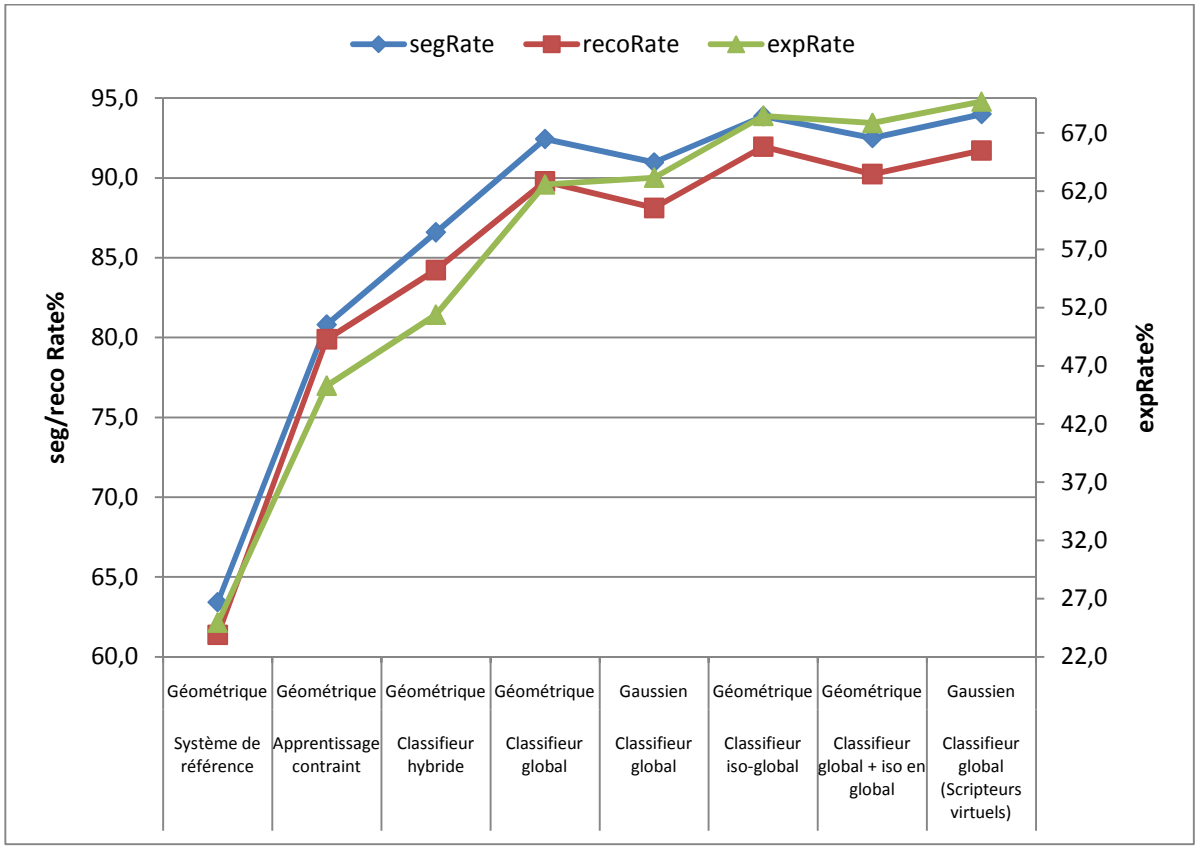



Figure 78 - Evolution des performances du reconnaisseur d'expressions (moyenne sur toutes les bases de tests)

Le Tableau 33 montre les performances réalisées par ce reconnaisseur sur les trois bases de test. La dernière ligne correspond à la moyenne pondérée de ces performances suivant la taille en nombre de symboles (pour segRate et recoRate) ou en nombre d'expressions de chaque base (pour expRate).

Tableau 33 - Performance du meilleur reconnaisseur d'expressions obtenu

| Base de test | segRate | recoRate | expRate |
|---|---|---|---|
| RamanReduced_CIEL | 94,3 | 92,1 | 71 |
| RamanReduced_Réelle | 86 | 81,4 | 38,6 |
| RamanReduced_Wiki_CIEL | 89,5 | 86,5 | 52,6 |
| **Moyenne** | **94** | **91,7** | **69,4** |

Dans une application interactive de saisie d'expressions mathématiques, il sera intéressant de prendre en compte plusieurs solutions du reconnaisseur d'expressions. En effet, comme l'illustre la Figure 79, la bonne solution n'est pas nécessairement en première position du système. Nous pouvons constater que le taux de reconnaissance d'expressions s'améliore en considérant les 5 premiers résultats du reconnaisseur de 71% à 77% sur la base RamanReduced_CIEL (38.6% à 50% et 52.6% à 65% sur les deux bases réelles).

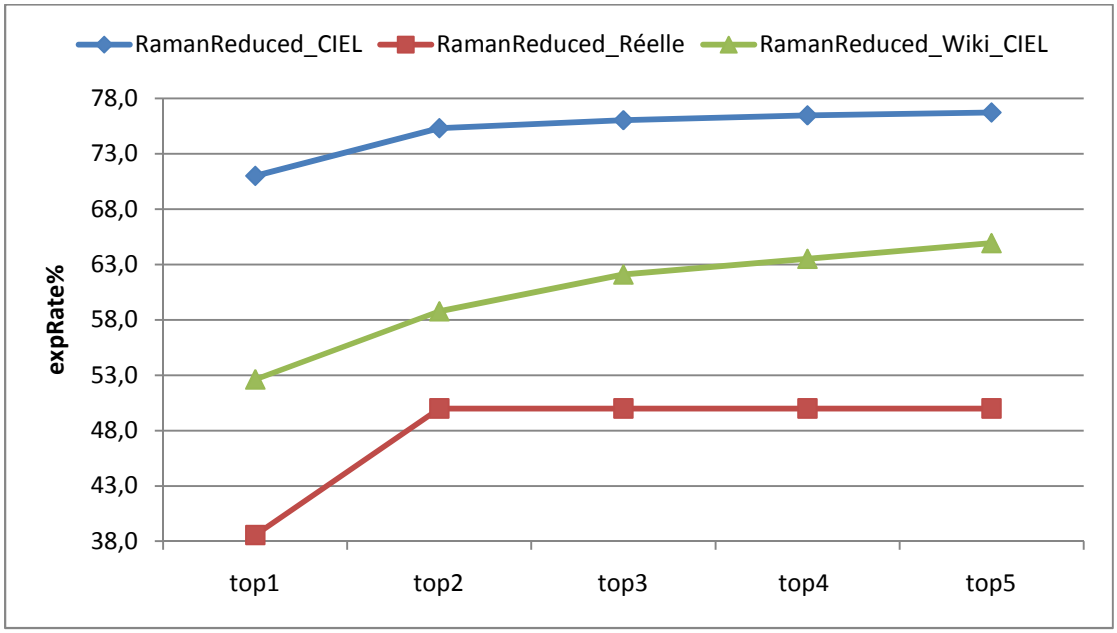



Figure 79 - Taux de reconnaissance d'expression en considérant les 5 premiers résultats

Comme nous l'avons vu, la comparaison directe avec d'autres résultats n'est pas appropriée. Néanmoins, nous résumons dans le Tableau 34 quelques résultats, en plus des nôtres.

Tableau 34 - Résumé des performances dans l'état de l'art

| Système | #Expression | segRate | recoRate | expRate | Remarques |
|---|---|---|---|---|---|
| Notre système | 281 | 88,3 | 84,7 | 49,1 | Taux moyen sur toutes les bases réelles de test |
| Notre système | 3881 | 94 | 91,7 | 69,4 | Taux moyen sur toutes les bases de test (synthétiques et réelles) |
| (Kim et al. 2009) [**80**] | 1500 | x | x | 58,7 | |
| (Rhee et al. 2009) [**15**] | | 94,8 | 84,8 | 29,2 | Taux moyen sur les deux base KME-I et KME-II |
| (Chan et al. 2001) [**78**] | 600 | x | 99,4 | 88,7 | Utilise un module supplémentaire pour la détection et la correction d'erreurs |
| (Fitzgerald et l. 2007) [**68**] | | x | x | 77,6 | |
| (Toyozumi et al. 2001) [**109**] | | x | 80 | x | |
| (Prusa et al. 2007) [**19**] | 330 | x | x | 85,5 | |
| (Fukuda et al. 1999) [**23**] | 160 | 97,2 | 99,3 | x | |
| (Geneo et al. 2006) [**67**] | 60 | x | x | 88,3 | |
| (Yamamoto et al. 2006) [**24**] | | 99,4 | 92,8 | x | |
| (Shi et al. 2007) [**97**] | | x | 96,6 | x | |

Pour affiner la compréhension de ce résultat, nous présentons dans l'annexe C les résultats détaillés obtenus de ce meilleur reconnaisseur d'expressions.

## 5.10. Applications

Au delà du développement des travaux académiques, nous avons voulu pouvoir disposer d'outils de démonstrations pour permettre à un utilisateur de tester le comportement de notre système en situation de saisie. Pour cela, nous avons implémenté deux applications dédiées à des problèmes différents dans le contexte de reconnaissance d'expressions mathématiques manuscrites. Il faut toutefois noter que la conception et le développement de telles interfaces interactives s'inscrivent dans un domaine vaste et actif de recherche : les Interfaces Homme Machine (IHM) dont nous ne sommes pas des spécialistes. Ces réalisations ne sont donc qu'une première ébauche. Néanmoins, les applications ont été testées à la fois par des spécialistes du domaine de la reconnaissance de l'écriture, des scientifiques et des étudiants venant d'autres domaines de recherches. Malgré quelques difficultés face aux nouvelles situations de saisies (tableau blanc interactif, tablette PC, iPod, …), les résultats de reconnaissance ont été perçus de façon subjective de manière très satisfaisante. Là encore, il serait souhaitable de pourvoir mettre en place des protocoles de tests pour permettre de juger de la satisfaction de l'usager.

### 5.10.1. Une calculatrice sur iPod

Cette application permet de faire des opérations mathématiques simples grâce à l'écran tactile de l'iPod. Elle a été réalisée par des étudiants au cours d'un projet de Master. L'idée est de remplacer la calculatrice traditionnelle à boutons en introduisant une interface capable d'appliquer les opérations élémentaires (+, -, ×, ÷) sur des valeurs numériques dessinées au doigt sur l'écran. Plusieurs opérations peuvent être appliquées séquentiellement. Il suffit juste de saisir les valeurs séparées par les opérateurs désirés. La saisie se termine par la saisie du signe '=' afin de calculer le résultat des opérations. Nous voyons qu'il s'agit d'un problème en une dimension où les symboles sont écrits de gauche à droite.

Le reconnaisseur d'expressions est basé sur une approche classique de segmentation, car le générateur des hypothèses (et le parseur) fourni par notre partenaire industriel ne sont pas compatible pour le développement sur une plateforme Apple. Donc, la segmentation et la reconnaissance sont faites à la volée par des heuristiques simples. Dès qu'un (ou plusieurs) symbole(s) est saisi une segmentation basé sur des projections sur l'axe X est faite. Ensuite un classifieur (entraîné préalablement sur la base Calculette isolée) est utilisé pour associer chacune des segmentations à l'une des 15 classes (chiffres ou opération). Finalement, lorsque le signe '=' est rencontré l'expression est évaluée afin de trouver le résultat du calcul. La Figure 80 montre une capture d'écran de l'interface avec quelques détails de son fonctionnement. Cette application a été développée en Objective C (Cocoa) avec le SDK d'Apple dédié à l'iPod.

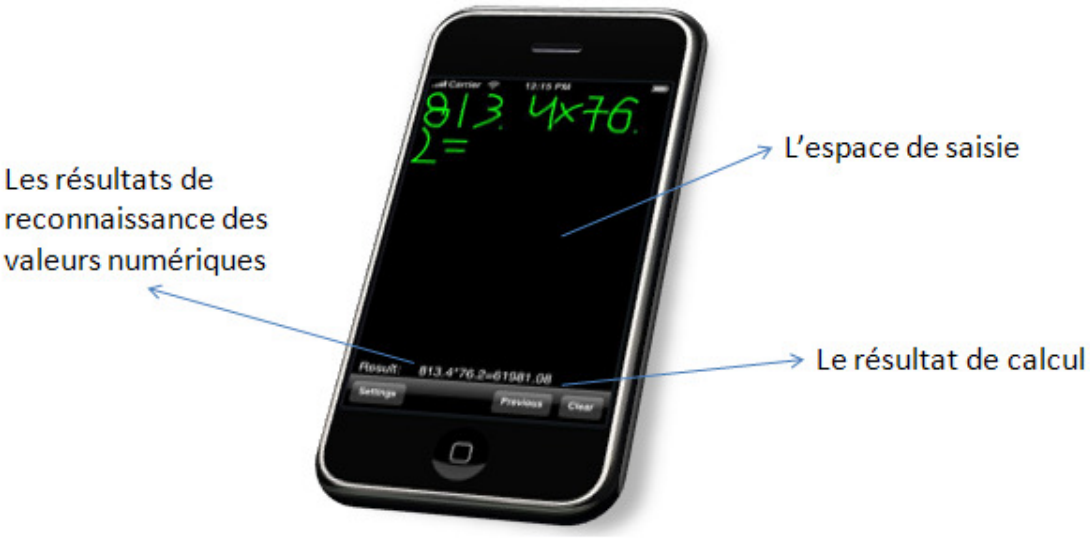



Figure 80 - Une capture d'écran de l'application calculatrice

### 5.10.2. Une interface pour saisir des expressions mathématiques manuscrites

Le second prototype fonctionne dans l'environnement PC, il est destiné à la saisie d'expressions mathématiques complexes. Aucune contrainte n'est imposée à l'utilisateur, il faut juste que les symboles utilisés soient compris dans le vocabulaire du classifieur (RamanReduced, 34 symboles). L'interface est basée sur l'architecture globale que nous avons proposée. L'utilisateur est invité à écrire son expression en toute liberté avant d'appuyer sur un bouton indiquant la fin de saisie. Ensuite l'encre obtenue est envoyée au système où toute la chaîne de la reconnaissance d'expressions est appliquée. Le résultat est finalement affiché en format LaTeX, et aussi converti en image. L'utilisateur peut ensuite compléter son expression ou en saisir une nouvelle. La Figure 81 montre une capture de l'interface détaillant ses fonctionnalités. Cette application a été développée avec Visual Studio.

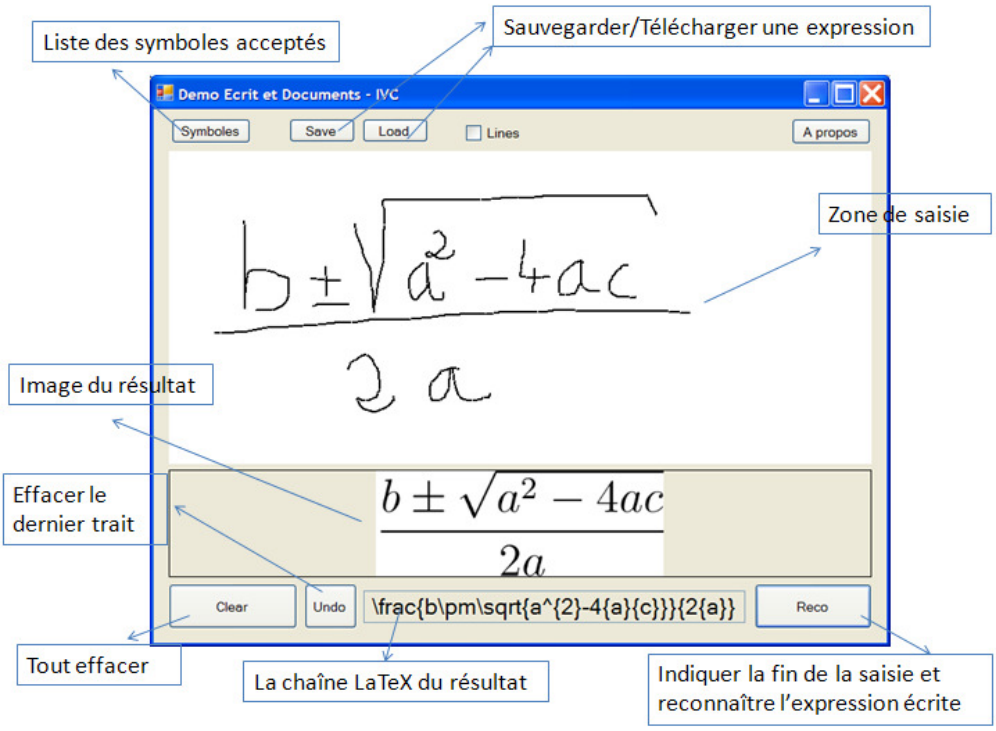



Figure 81 - Une capture d'écran de l'interface de saisie d'expressions mathématiques

# CONCLUSION ET PERSPECTIVES

Le but des travaux menés au sein de cette thèse a été de proposer une contribution au domaine de la reconnaissance des langages bidimensionnels manuscrits en-ligne. En plus de l'ambiguïté qui se trouve naturellement dans les documents manuscrits, ces langages portent plus de complexité en raison de leur nature bidimensionnelle. Cela rend les trois phases classiques de la reconnaissance de l'écriture (la segmentation, la reconnaissance des symboles, l'interprétation) encore plus difficile. Le positionnement des symboles n'est plus systématiquement de gauche à droite (ou l'inverse dans certains scripts), mais les symboles peuvent s'écrire dans toutes les directions. Les techniques de segmentation doivent donc respecter cette propriété. La phase de reconnaissance des symboles est un problème classique de reconnaissance de formes. Toutefois, du fait du nombre important de classes présentes (plus de 220 symboles pour les expressions mathématiques) la difficulté s'en trouve accentuée. La dernière phase, l'interprétation, est la plus difficile car il s'agit dans un premier temps d'analyser la structure 2D du document avant d'appliquer une analyse syntaxique adaptée aux langages 2D.

Après avoir exploré l'état de l'art de ce domaine nous avons concentré plus particulièrement notre étude sur la reconnaissance d'expressions mathématiques manuscrites en-ligne. L'absence de bases de données publiques est une grande difficulté à laquelle sont confrontés les chercheurs de ce domaine. Dans cette optique, nous avons développé l'outil LaTeX2Ink qui permet de générer des grosses bases d'expressions mathématiques à partir des chaînes LaTeX et d'une base de symboles isolés. Ces bases, dites synthétiques, ont été dédiées à l'apprentissage et au réglage du système. Pour rendre l'évaluation davantage réaliste nous avons collecté deux bases, dites réelles, d'expressions mathématiques. Les mesures utilisées pour évaluer la performance de notre système sont : le taux de segmentation, le taux de reconnaissance des symboles, le taux de reconnaissance d'expressions. Ces mesures sont les plus utilisées dans la littérature. De plus, nous avons proposé de nouvelles mesures pour évaluer la reconnaissance des relations spatiales. Pour l'instant, nous n'avons mis en œuvre qu'une version expérimentale limitée de ces mesures car leurs calculs nécessitent de mettre au point des algorithmes complexes de comparaison d'arbres.

La reconnaissance de structures 2D se déroule classiquement en trois étapes séquentielles : la segmentation, la reconnaissance, et l'interprétation. Plus récemment, plusieurs travaux considèrent une segmentation et reconnaissance des symboles simultanées. Nous avons proposé, à l'instar d'autres travaux récents, un système basé sur une approche globale qui effectue une optimisation simultanée de ces trois étapes. Nous transférons le problème de reconnaissance en la recherche de la meilleure interprétation possible d'un ensemble de traits d'entrée. Contrairement à beaucoup de travaux existants, nous avons considéré un classifieur de symboles doté d'une capacité de rejet pour gérer les segmentations invalides proposées par un générateur d'hypothèses de symboles. Nous avons présenté deux stratégies pour inclure le rejet dans le classifieur. La première est

de combiner deux classifieurs (classifieur de symboles et classifieur du rejet) dans un classifieur hybride. Dans ce cas, le classifieur de rejet est en charge de distinguer entre les bonnes et mauvaises segmentations. La seconde solution, plus compacte et toute aussi performante, est d'ajouter une sortie additionnelle au classifieur pour qu'il représente la classe de rejet.

Une originalité de notre système réside dans le schéma d'apprentissage global du système. Cet apprentissage permet d'entraîner le classifieur de symboles directement à partir des expressions complètes au lieu d'utiliser un classifieur appris en isolé. L'avantage de ce schéma est de pouvoir avoir des exemples de rejet dans le contexte de reconnaissance d'expressions pour apprendre cette classe. Nous avons proposé plusieurs stratégies pour réaliser cet apprentissage et enrichir la connaissance du classifieur. En effet, le classifieur peut être initialisé par un apprentissage isolé avant d'appliquer un apprentissage global (apprentissage iso-global). De plus, la base de symboles isolés peut être considérée comme une base d'expressions à un seul symbole qui peut participer à l'apprentissage global pour l'enrichir encore plus. Nous avons également proposé une modélisation contextuelle basée sur l'analyse structurelle de l'expression. Cette analyse structurelle s'effectue à base de règles empiriques ou encore en se basant sur des modèles probabilistes appris lors d'un apprentissage sur une base d'expressions. Les modèles appris permettent d'éviter les problèmes de situations imprévisibles entre les symboles d'une expression en modélisant ces situations telles qu'elles apparaissent dans les expressions.

Finalement, nous avons présenté et analysé de nombreuses expérimentations et les résultats obtenus ont permis de situer les différentes approches proposées. Ainsi, le taux de reconnaissance d'expressions a évolué de 24% en utilisant un classifieur appris en isolé sans capacité de rejet à 70% en utilisant la meilleure configuration du système. Cette configuration consiste en un classifieur global avec une classe explicite de rejet et une modélisation probabiliste des relations spatiales. Nous avons également adapté le système actuel à la reconnaissance d'organigrammes manuscrites en-ligne avec un succès acceptable compte tenu de l'effort minimal d'adaptation réalisé.

***Perspectives***

Nous avons souligné tout au long de ce manuscrit plusieurs points qui pourraient améliorer le système global. Le point le plus important à aborder va être l'apprentissage global du système à partir d'une base d'expressions réelles. Car même si le générateur d'expressions fournit de grandes quantités d'expressions, il a montré ses limites surtout en ce qui concerne l'apprentissage des modèles de relations spatiales. De plus, il va falloir impérativement étendre le corpus d'expressions utilisées en apprentissage afin de considérer davantage de symboles et de relations. L'utilisation d'une base réelle nécessite un travail supplémentaire pour étiqueter la vérité terrain des expressions. Bien entendu, cette tâche est longue et fastidieuse. Une autre extension serait de pouvoir se passer de cet étiquetage pendant l'apprentissage. Dans cette même optique, une thèse intitulée « Extraction de connaissances symboliques et relationnelles appliquée aux tracés manuscrits structurés en-ligne » a démarré en 2009. L'idée principale de cette thèse est de travailler à l'automatisation de l'apprentissage d'un système de reconnaissance de documents structurés complexes (langages 2D) en évitant l'étiquetage manuel d'une grande quantité de données d'apprentissage.

En ouvrant sur des horizons plus larges ces travaux, nous pouvons les mettre en perspective pour la réalisation d'un système de reconnaissance multi-modal. En effet, l'écriture manuscrite et la parole constituent les deux modes d'interaction les plus naturels dont sont dotés les êtres humains pour communiquer. Chacune de ces modalités possède des spécificités propres aussi bien du point de vue des usages, de leur expressivité que des outils et techniques de numérisation qui leur sont associés. Dans cette optique, une thèse intitulée « Stratégies de fusion pour des signaux écrits et sonores – Application à la reconnaissance d'expressions mathématiques » vient de démarrer. L'objectif de cette thèse est d'étudier les stratégies de fusion pour un système multimodal écriture dynamique (en-ligne)/parole, qui permettent soit d'étendre soit de surpasser les performances des systèmes utilisant une seule des deux modalités.

L'objectif ultime de nos travaux est de concevoir un système générique de reconnaissance de langages 2D tel que les spécificités du langage soient extraites par apprentissage et le moins possible formulées a priori.

# ANNEXES

# Annexe A – Analyse lexicale et syntaxique dans LaTeX2Ink

## 1. Liste de lexèmes considérés par le générateur LaTeX2Ink

Flex effectue l'analyse lexicale de la chaîne LaTeX en entrée. Il renvoie alors un lexème (ex : RO, AN, FRAC) correspondant à la classe reconnue. Nous avons défini 23 classes :

– **AN** : Chiffre arabe ;

– **RL** : Lettre romaine minuscule ;

– **RLa** : Lettre romaine minuscule ascendante ;

– **RLd** : Lettre romaine minuscule descendante ;

– **RLad** : Lettre romaine minuscule ascendante et descendante ;

– **RU** : Lettre romaine majuscule ;

– **MO** : Opérateur mathématique ;

– **RO** : Opérateur relationnel ;

– **AS** : Flèche ;

– **FW** : Fonction ;

– **FWa** : Fonction ascendante ;

– **FWd** : Fonction descendante ;

– **FWad** : Fonction ascendante et descendante ;

– **GS** : Symbole grec ;

– **ES** : Symbole élastique ;

– **MS** : Symbole divers ;

– **FRAC** : Fraction ;

– **SQRT** : Racine carrée ;

– **BAR** : Symbole barre ;

– **HAT** : Symbole chapeau ;

– **BR** : Délimiteur ;

– **EXP** : Exposant ;

– **IND** : Indice

## 2. La grammaire utilisée par Bison pour générer l'analyseur syntaxique du générateur LaTeX2Ink

L'analyse ascendante est basée sur une grammaire hors contexte :

```
S : dollar EXPR dollar
EXPR : ELE | ELE EXPR
ELE : IND ACCG F ACCD
        | EXP ACCG EXPR ACCD
        | EXP ACCG EXPR ACCD IND ACCG EXPR ACCD
        | FRAC ACCG EXPR ACCD ACCG EXPR ACCD
        | SQRT ACCG EXPR ACCD
        | BAR ACCG EXPR ACCD
        | HAT ACCG EXPR ACCD
        | AN
        | RU
        | RL
        | RLa
        | RLd
        | RLad
        | ACCG
        | ACCD
        | MO
        | MSs
        | GS
        | FW
        | FWa
        | FWd
        | FWad
        | RO
        | BR
        | LIM
        | LIM IND ACCG F ACCD
        | ES
        | ES EXP ACCG F ACCD
        | ES IND ACCG F ACCD
        | ES IND ACCG F ACCD EXP ACCG F ACCD
```

La première règle est appelée axiome, elle permet de vérifier que la chaîne en entrée est une expression LaTEX valide. En effet les expressions mathématiques en LaTEX sont reconnues grâce au délimiteur « $ ».La deuxième règle sert à effectuer l'analyse de la chaîne de gauche à droite, ainsi une expression est constituée soit d'un élément, soit d'un élément suivi d'une expression. La troisième règle permet de définir tous les types de structures et d'éléments que l'on peut trouver dans une expression LaTEX. Par exemple les structures de type exposant, indice... ou des éléments comme les lettres romaines minuscules, les symboles grecs...

# Annexe B - Informations supplémentaires des bases de données

## 1. Les symboles mathématiques considérés dans la base isolée

L'Alphabet latin : a...z, A...Z ; Les chiffres : 0…9

Le reste des symboles est organisé dans le tableau suivant :

| Alphabet grec | | Opérateurs binaires | | Symboles élastiques | | Fonctions | | Flèches | |
|---|---|---|---|---|---|---|---|---|---|
| **Nom** | **Symboles** | **Nom** | **Symboles** | **Nom** | **Symboles** | **Nom** | **Symboles** | **Nom** | **Symboles** |
| Gamma | $\Gamma$ | + | $+$ | sum | $\sum$ | cos | cos | leftarrow | $\leftarrow$ |
| Delta | $\Delta$ | - | $-$ | prod | $\prod$ | sin | sin | Leftarrow | $\Leftarrow$ |
| Theta | $\Theta$ | * | $\times$ | coprod | $\coprod$ | tan | tan | rightarrow | $\rightarrow$ |
| Lambda | $\Lambda$ | div0 | $\div$ | int | $\int$ | arccos | arccos | Rightarrow | $\Rightarrow$ |
| Phi | $\Phi$ | / | $/$ | iint | $\iint$ | arcsin | arcsin | leftrightarrow | $\leftrightarrow$ |
| Psi | $\Psi$ | backslash | $\backslash$ | iiint | $\iiint$ | arctan | arctan | Leftrightarrow | $\Leftrightarrow$ |
| Omega | $\Omega$ | = | $=$ | oint | $\oint$ | sinh | sinh | uparrow | $\uparrow$ |
| alpha | $\alpha$ | pm | $\pm$ | oiint | $\oiint$ | cosh | cosh | Uparrow | $\Uparrow$ |
| beta | $\beta$ | mp | $\mp$ | oiiint | $\oiiint$ | tanh | tanh | downarrow | $\downarrow$ |
| gamma | $\gamma$ | < | $<$ | sqrt | $\sqrt{}$ | min | min | Downarrow | $\Downarrow$ |
| delta | $\delta$ | > | $>$ | vline | $\vert$ | max | max | updownarrow | $\updownarrow$ |
| epsilon | $\varepsilon$ | neq | $\neq$ | { | { | exp | exp | Updownarrow | $\Updownarrow$ |
| zeta | $\zeta$ | equiv | $\equiv$ | } | } | log | log | mapsto | $\mapsto$ |
| eta | $\eta$ | nequiv | $\not\equiv$ | [ | [ | ln | ln | downleftarrow | $\hookleftarrow$ |
| theta | $\theta$ | leq | $\leq$ | ] | ] | lim | lim | | |
| iota | $\iota$ | geq | $\geq$ | ( | ( | arg | arg | | |
| kappa | $\kappa$ | sim | $\sim$ | ) | ) | deg | deg | | |
| lambda | $\lambda$ | simeq | $\cong$ | infty | $\infty$ | rad | rad | | |
| mu | $\mu$ | approx | $\approx$ | nabla | $\nabla$ | dim | dim | | |
| nu | $\nu$ | ll | $<<$ | angle | $\angle$ | inf | inf | | |
| xi | $\xi$ | gg | $>>$ | prime | $'$ | sup | sup | | |
| pi | $\pi$ | in | $\in$ | partial | $\partial$ | Pr | Pr | | |
| rho | $\rho$ | notin | $\notin$ | ldots | $\ldots$ | det | det | | |
| sigma | $\sigma$ | ni | $\ni$ | forall | $\forall$ | amp | amp | | |
| tau | $\tau$ | notni | $\not\ni$ | exists | $\exists$ | div | div | | |
| upsilon | $\upsilon$ | subset | $\subset$ | nexists | $\nexists$ | rot | rot | | |
| phi | $\varphi$ | nsubset | $\not\subset$ | varnothing | $\varnothing$ | Im | $\Im$ | | |
| chi | $\chi$ | subseteq | $\subseteq$ | circ | $\circ$ | Re | $\Re$ | | |
| psi | $\psi$ | nsubseteq | $\not\subseteq$ | bullet | $\bullet$ | Fourier | $F$ | | |
| omega | $\omega$ | supset | $\supset$ | neg | $\neg$ | Laplace | $L$ | | |
| | | nsupset | $\not\supset$ | hat | $\hat{}$ | Natural | $N$ | | |
| | | supseteq | $\supseteq$ | ast | * | Relative | $Z$ | | |
| | | nsupseteq | $\not\supseteq$ | % | $\%$ | Rational | $Q$ | | |
| | | perp | $\perp$ | . | $\cdot$ | Real | $\Re$ | | |
| | | parallel | $\parallel$ | , | $,$ | Complex | $C$ | | |
| | | oplus | $\oplus$ | therefore | $\therefore$ | | | | |
| | | otimes | $\otimes$ | because | $\because$ | | | | |
| | | ominus | $\ominus$ | | | | | | |
| | | odot | $\odot$ | | | | | | |
| | | cap | $\cap$ | | | | | | |
| | | cup | $\cup$ | | | | | | |
| | | wedge | $\wedge$ | | | | | | |
| | | vee | $\vee$ | | | | | | |
| | | propto | $\propto$ | | | | | | |
| | | ! | ! | | | | | | |

## 3. Les 36 expressions du corpus RamanReduced

| Chaînes LaTeX | Expressions | #symboles |
|---|---|---|
| $a+b+c+d$ | $a + b + c + d$ | 7 |
| $a+\frac{b}{c} +d$ | $a + \frac{b}{c} + d$ | 7 |
| $\frac{a+b}{c+d}$ | $\frac{a + b}{c + d}$ | 7 |
| $\frac{a}{b}+c+d$ | $\frac{a}{b} + c + d$ | 7 |
| $\frac{a}{b+c+d}$ | $\frac{a}{b + c + d}$ | 7 |
| $a+\frac{b+c}{d+e}+x$ | $a + \frac{b + c}{d + e} + x$ | 11 |
| $a+bc+d$ | $a + bc + d$ | 6 |
| $(a+b)(c+d)$ | $( a + b )( c + d )$ | 10 |
| $\frac{x+y^{2}}{k+1}$ | $\frac{x + y^{2}}{k + 1}$ | 8 |
| $\frac{x+y^{2}}{ k}+1$ | $\frac{x + y^{2}}{k} + 1$ | 8 |
| $x+\frac{y^{2}}{k+1}$ | $x + \frac{y^{2}}{k + 1}$ | 8 |
| $x+\frac{y^{2}}{ k}+1$ | $x+\frac{y^{2}}{k}+1$ | 8 |
| $x+y^{\frac{2}{ k+1}}$ | $x+y^{\frac{2}{k+1}}$ | 8 |
| $(x^{2})^{y} = x^{2y}$ | $(x^{2})^{y} = x^{2y}$ | 9 |
| $1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\ldots}}}}}}$ | $1 + \frac{x}{1 + \frac{x}{1 + \frac{x}{1 + \frac{x}{1 + \frac{x}{1 + \ldots}}}}}$ | 23 |
| $(a+b)^{3}=a^{3}+3a^{2}b+3ab^{2}+b^{3}$ | $(a+b)^{3} = a^{3} + 3a^{2}b + 3ab^{2} + b^{3}$ | 22 |
| $a^{3}+b^{3}=(a+b)(a^{2}-ab+b^{2})$ | $a^{3} + b^{3} = (a+b)(a^{2} - ab + b^{2})$ | 21 |
| $x= \frac{-b \pm \sqrt{b^{2}-4ac}}{2a}$ | $x = \frac{-b \pm \sqrt{b^{2} - 4ac}}{2a}$ | 16 |
| $\frac{1+\sqrt{5}}{2}=\phi$ | $\frac{1+\sqrt{5}}{2} = \varphi$ | 8 |
| $\frac{\sqrt{\pi}}{2} \neq \sqrt{\frac{\pi}{2}}$ | $\frac{\sqrt{\pi}}{2} \neq \sqrt{\frac{\pi}{2}}$ | 9 |
| $\sin^{2}x+\cos^{2}x=1$ | $sin^{2} x + cos^{2} x = 1$ | 9 |
| $\sin x^{2} + \cos x^{2} \neq 1$ | $sinx^{2} + cosx^{2} \neq 1$ | 9 |

| | | |
|---|---|---|
| $\sin (a+b) = \sin a \cos b + \cos a \sin b$ | $sin(a+b) = sin a cos b + cos a sin b$ | 16 |
| $\cos (x+y)=\cos x \cos y - \sin x \sin y$ | $cos(x+y) = cos x cos y - sin x sin y$ | 16 |
| $\sin 2x = 2 \sin x \cos x $ | $sin\ 2x = 2\ sin\ x cos\ x$ | 9 |
| $\cos 2x = \cos^{2} x -\sin^{2} x$ | $cos 2x = cos^2 x - sin^2 x$ | 11 |
| $\log^{2}x\neq2\log x$ | $log^2 x \neq 2logx$ | 7 |
| $\log x^{2}=2\log x$ | $logx^2 = 2logx$ | 7 |
| $\frac{\log x}{\log a} = \log_{a} x$ | $\frac{log x}{log a} = log_a x$ | 9 |
| $1+x+\frac{x^{2}}{2}+\frac{x^{3}}{3}+\cdots +\frac{x^{n}}{n}+\cdots$ | $1+\frac{x^2}{2}+\frac{x^3}{3}+\cdots+\frac{x^n}{n}+\cdots$ | 20 |
| $ x - \frac{x^{2}}{2} +\frac{x^{3}}{3}-\frac{x^{4}}{4}+\frac{x^{5}}{5} \pm \cdots  = \log(1+x)$ | $x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}+\frac{x^5}{5}\pm\cdots = log(1+x)$ | 30 |
| $\gamma = 1+\frac{1}{2}+\frac{1}{3} +\frac{1}{4} +\cdots +\frac{1}{n}-\log n $ | $\gamma = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n}-log\ n$ | 24 |
| $\int\frac{dx}{x} =\log x$ | $\int\frac{dx}{x} = log\ x$ | 8 |
| $\sum_{i=1}^{n} a_{i} =1$ | $\sum_{i=1}^{n} a_i = 1$ | 9 |
| $\sum_{i=1}^{n} a_{i} + b_{i} = 1$ | $\sum_{i=1}^{n} a_i + b_i = 1$ | 12 |
| $\cosh^{2}x -\sinh^{2} x = 1$ | $cosh^2 x - sinh^2 x = 1$ | 9 |

4. Fréquence des symboles dans le corpus RamanReduced

| Symbole | Fréquence |
|---|---|
| 1 | 26 |
| 2 | 36 |
| 3 | 12 |
| 4 | 4 |
| 5 | 3 |
| a | 25 |
| b | 22 |
| c | 9 |
| d | 9 |
| e | 1 |
| i | 5 |
| k | 5 |
| n | 6 |
| x | 48 |
| y | 10 |
| + | 63 |
| - | 41 |
| = | 20 |
| pm | 2 |
| neq | 3 |
| gamma | 1 |
| pi | 2 |
| phi | 1 |
| sum | 2 |
| int | 1 |
| sqrt | 8 |
| ( | 9 |
| ) | 9 |
| ldots | 5 |
| cos | 10 |
| sin | 10 |
| sinh | 1 |
| cosh | 1 |
| log | 10 |

## 5. Le corpus RamanReduced_Wiki_CIEL

```
1 + 1 = 3
{i}^2 = -1
\frac{\pi}{2}
i^4 = 1
{\frac{\sqrt{3}}{2}}+{\frac{1}{2}}i
{\frac{-\sqrt{3}}{2}}+{\frac{1}{2}}i
\frac{1}{d^2}
\log_ca
e^{i\pi}= -1
\frac{a}{c}=\frac{c}{b}
(a_{n})
2 = \sqrt{2+2}
\cos(\pi) = -1
n(1)
\frac{\sqrt{4}}{2}
\frac{1+\sqrt{5}}{2}
1 + \frac 1{1 + \frac { 1}{1}}
2\sin(x)
n(2n+1)
\sin(1)
x_n=b
\frac{\pi-3}{\sqrt{2}}
\frac{3^{\frac{1}{2}}}{2}
3+2\sqrt{2}
i_1 + i_4 = i_2 + i_3
n=b-a+1
n(n+2)
2^{e} - 1
(n+1)^2
(a-b)^2 = a^2 - 2ab + b^2
2^{\sqrt{2}}
\sqrt{2}^{\sqrt{2}}
n^{(1)}
(a + b)^2 = a^2 +2ab + b^2
4\pi^2 a^3
e_2 = 21
e_3 = 15
i_{1}=4
x=1
i^4 = 1
\cos(\pi) = - 1
2^{43}
x \neq y
e^{ix}\neq 1
ax^2+bxy+cy^2
e^{-x}
\cosh(1) = \frac {e^2+1}{2e}
\cosh(i) = \cos(1)
-2ab \cos\gamma
2^{-1}
\sinh(1) = \frac {e^2-1}{2e}
\sinh(i) = i \sin(1)
ax^2 = c
\sqrt{\sqrt{n^2}}+1
\frac{2\pi}{3}
2i\pi
4_12_12
i^2 = -1
x^3 + 3(2 - \sqrt{3})x^2 - 3x - 2 + \sqrt{3}
x + x^2
\frac{5}{2} \sqrt{5+2\sqrt{5}}
(b_n + a_n)
222_1
{2 -\frac{\log(\sqrt{2})}{\log(2)}}
\frac{4}{3}\pi
{\frac{\sqrt{3}}{2}}+{\frac{1}{2}}i
\sin x=\frac{e^{ix}-e^{-ix}}{2i}
\frac{n(n-3)}{2}
-\frac{1}{x^2}
y^2=x^3-n^2x
c = \sum_{i=1}^{n} x_i a_i
1+i
b = ak
y = \frac{d}{a}
ab+ba
\frac{1}{b+1}
(n - 1)
\frac{n+2}{2}
y=-x^3-5x^2+\frac{2}{3}x-1
x^{(k)}
(2i)^n
x^4-x^2
x = \frac{2}{3}
\frac{2  d_2^2  (d_1+d_2-2)}{d_1  (d_2-2)^2
(d_2-4)}
(a+b)^2 = a^2 + 2ab + b^2
x+k
\sin(x)=\sin(y)
(2^n - 1)^2 - 2
anx^{n-1}
y = b
y^{(k)}
(a + b)^2 = a^2 +2ab + b^2
- \sin x
y = x^4+b_1x^3+b_2x^2+b_3x+b_4
c^2 = \frac{1}{2}d^2
\gamma_{12}
\frac{1 + \sqrt{5}}{2}
1 - \frac {1 }{ 3} = \frac {2 }{ 3}
\frac {1 + \frac { 1}{4}x}{1 - \frac { 3}{4}x+
\frac { 1}{4}x^2 - \frac 1{24}x^3}
3-n = 1
\pi(x) = \pi(y)
x^2=ax+b
(a + bi) (a - bi) = a^2 + b^2
2+\sqrt{2}
x^2=2x+1
y^{n-1}
\frac{1}{\sqrt{x^2-1}}
n_1+\cdots+n_{k}=n
\frac{-1}{n+1}
b_i(k)
\log_aa = 1
x = \sum_{i=1}^k a_i e_i
\sqrt{1+x^2}
a^2 + 2ab + b^2 - 2ab
-\sqrt y
\sqrt{(5-x)d}
x = a
\frac{b-a}{2^{n+1}}
y =x^2
x=x_{c}+y
2^{-1}
\frac 1{1 - x + \frac {x^2}2}
x_4 = \sqrt{2}
\frac{4}{3}\pi a b^2
k=n
x = \frac{1}{3}
2(2-1)=(1+1)(2-1)
(x_n)
```

```
\frac{1}{24}\pi^4
2 = 1
2=1+1
(a+b)(a-b)
1 + \sqrt{2}
d = 1
i_1 i_2=i_3
\sqrt{-1}
i = \sqrt{-1}
(c_n)
e^{i\pi}= -1
(-i)^2 = -1
4c = 2\sqrt2 d
2 = \frac {(1 + 3) }{ 2}
2^{2^{2^2}}
\sin(\frac{\pi}{3})
\frac{n(n+1)}{2}
a^2 + b^2 + c^2 + d^2 = 1
\frac{1}{2}\pi^2
x^3=x+1
a(e-a^{-1}(a-x))=x
3 + 2 = \log_2{32}
a = \sqrt{5}b
2^e-1
x^3+ax+b
y=x^2+5x-3
\sqrt {14}
132
y_{1} = 55
y=ax^2+bx+c
c_n = (-1)^n \sum_{k=1}^{n-1}
\frac1{\sqrt{k(n-k)}}
\frac{n^2}{2}+\frac{n}{2}+\frac{n^2}{2}-
\frac{n}{2}
n = 1
{ c = \sqrt{a^2-b^2} }
a = 2
a^2+b^2+c^2=1
cn = 1
\frac{\frac {2}{1}}{\frac {3}{2}}=\frac{4}{3}
2^2 - 1
y_i = a x_i + b
\frac{\sqrt{3}}{3}
\frac{1}{n(n+1)(n+2)}
2^{2^1}+1
(a+b)^2=a^2+b^2+2ab
n+i
\sin(\frac x2)
y^2=x^3+ax+b
y = x^n
\log_ca
x+2
x^3=x^2+x+1
a^2 + 2ab + b^2 - 2ab
\frac{x}{c}
a_3=1
\frac cb
\cos(\pi) = - 1
a + b
x_1 = -2 + i
(2 - 3i)
4\pi^2 a^3
i_1 + i_2
ak+n
1 + a = 1
\sqrt{a + b\sqrt c} = d + e\sqrt c
\cos (a - b) = \cos a \cos b + \sin a \sin b
\frac{-1 + i\sqrt{3}}{2}
2^{2^{2^2}}
\frac{n}{d}
\sum\frac{(-1)^n}n
k_2 = k_1
x = \sum_{i=1}^k a_i e_i
2^{k - 1} + k
\frac{5}{2}
(a-d)^2+4bc
\sum_{i=1}^n (2i-1) = n^2
y = x^2 + \frac1x
2a = b
a^2 + b^2 + c^2
2^{n-1}
(a+b)^2=a^2+2ab+b^2
e^{inx}
a^{b^{c}}
a_n = 1
```

## 6. Formulaire de collecte de la base réelle Raman

| Un formulaire à copier | Les expressions manuscrites |
|---|---|

$$\frac{a}{b}+c+d$$

$$x_1^k+x_2^k+x_3^k+\ldots+x_n^k=0$$

$$x+_n y+_n z$$

$$x+y^{\frac{2}{k+1}}$$

$$a^3+b^3=(a+b)(a^2-ab+b^2)$$

$$\log_{a^2} x=\frac{1}{\log_x a^2}=\frac{1}{2\log_x a}=\frac{\log_a x}{2}$$

$$\sin 2x=2\sin x\cos x$$

$$\log(1+x)-\log(1-x)=\log\frac{1+x}{1-x}=\sum_{i=1}^{\infty}\frac{x^{2i-1}}{2i-1}$$

$$\sin^2 x+\cos^2 x=1$$

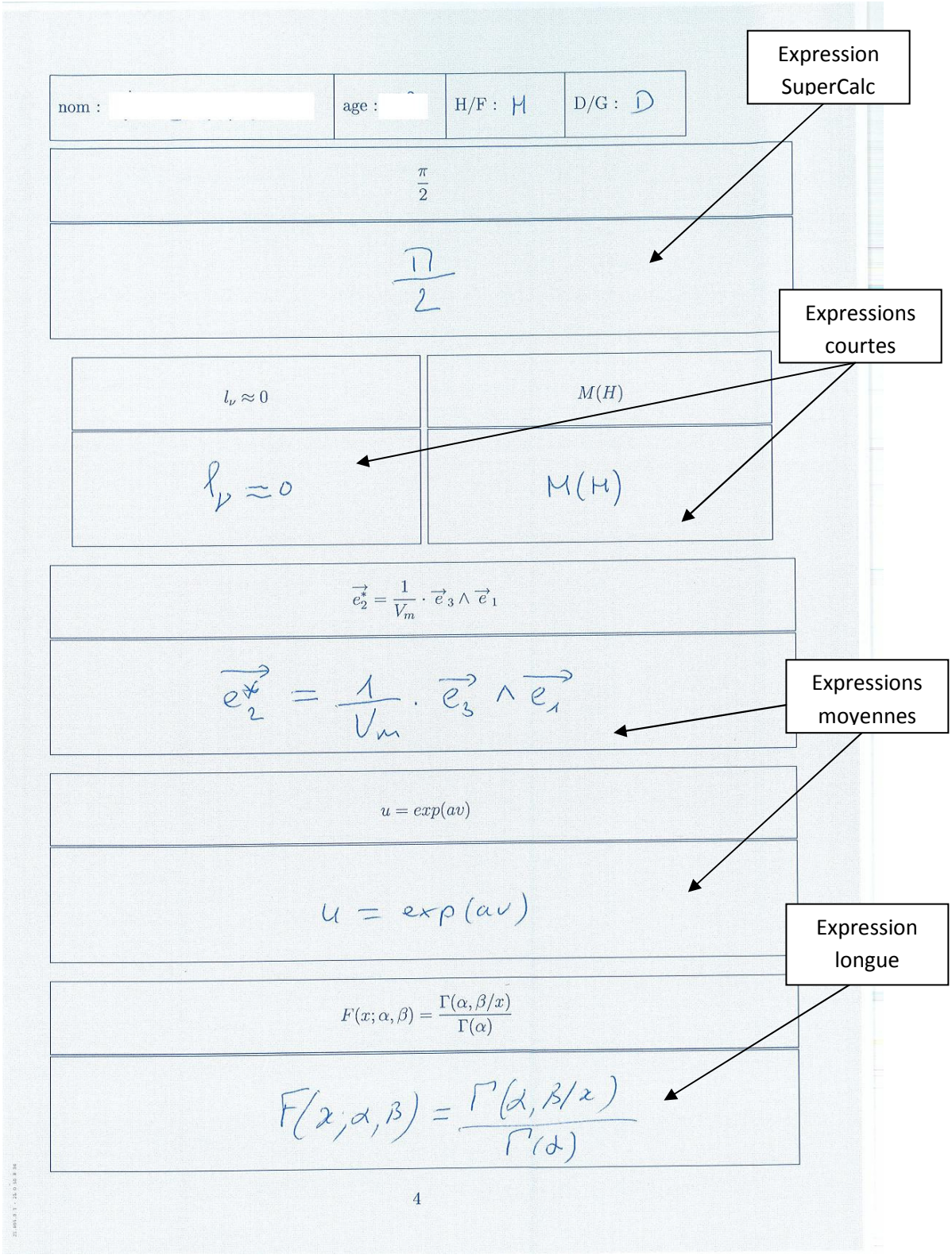
$$\int_0^1\int_0^{\sqrt{1-y^2}}1\,dx\,dy=\int_0^{\frac{\pi}{2}}\int_0^1 r\,dr\,d\theta$$

$$\lim_{x\to\infty}\int_0^x e^{-y^2}dy=\frac{\sqrt{\pi}}{2}$$

$$d(x,y)=\sqrt{(x_1-y_1)^2+(x_2-y_2)^2}$$

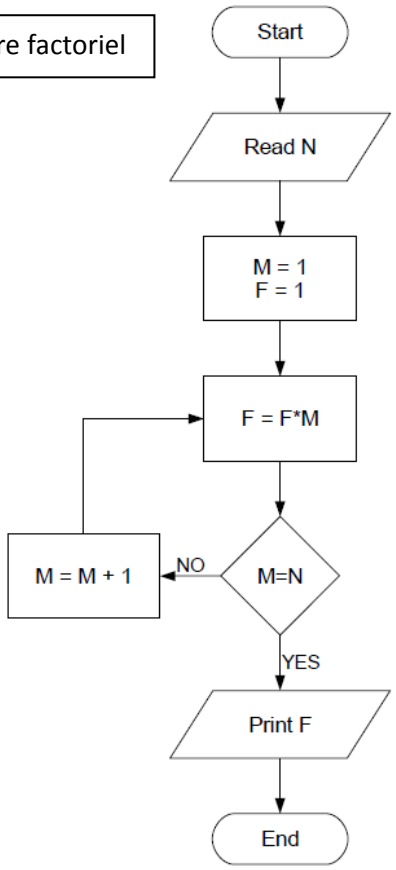
| | |
|---|---|
| Nom : | Gaucher ☐ Droitier ☐ |
| Age : | Homme ☐ Femme ☐ |

## 7. Formulaire de collecte de la base réelle Wiki_CIEL

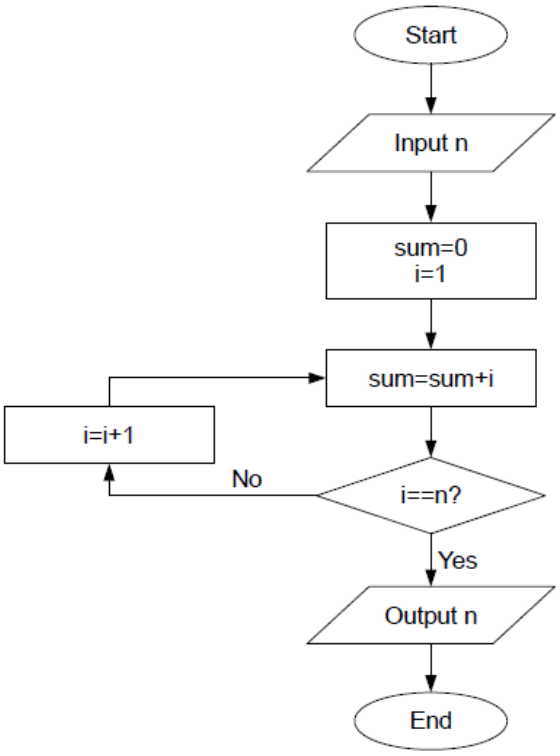


## 8. Liste des organigrammes

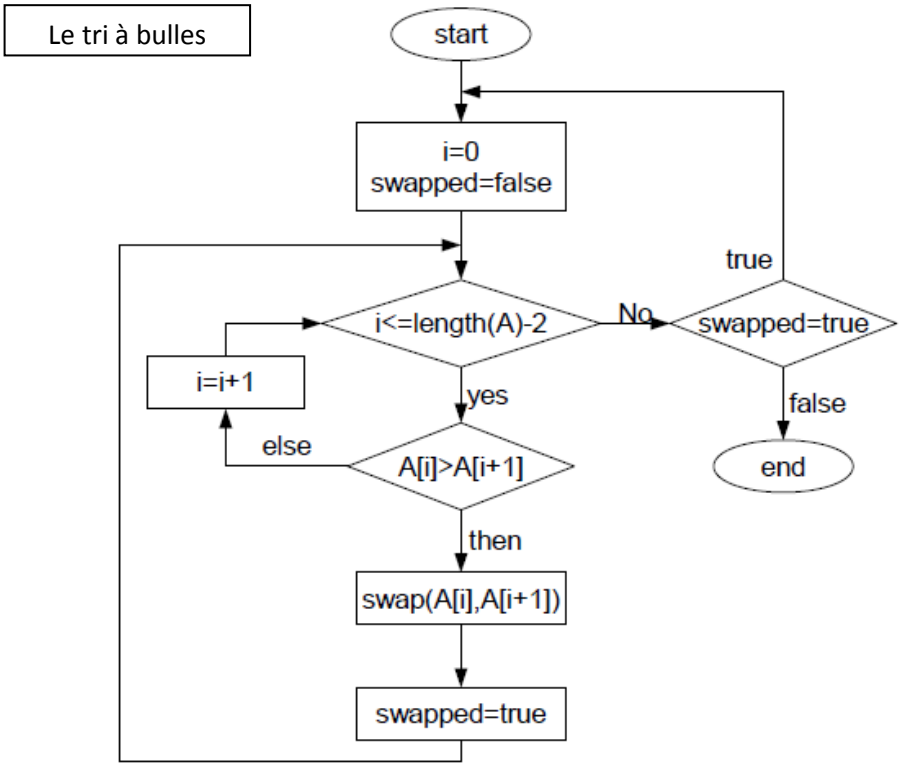
Nombre factoriel



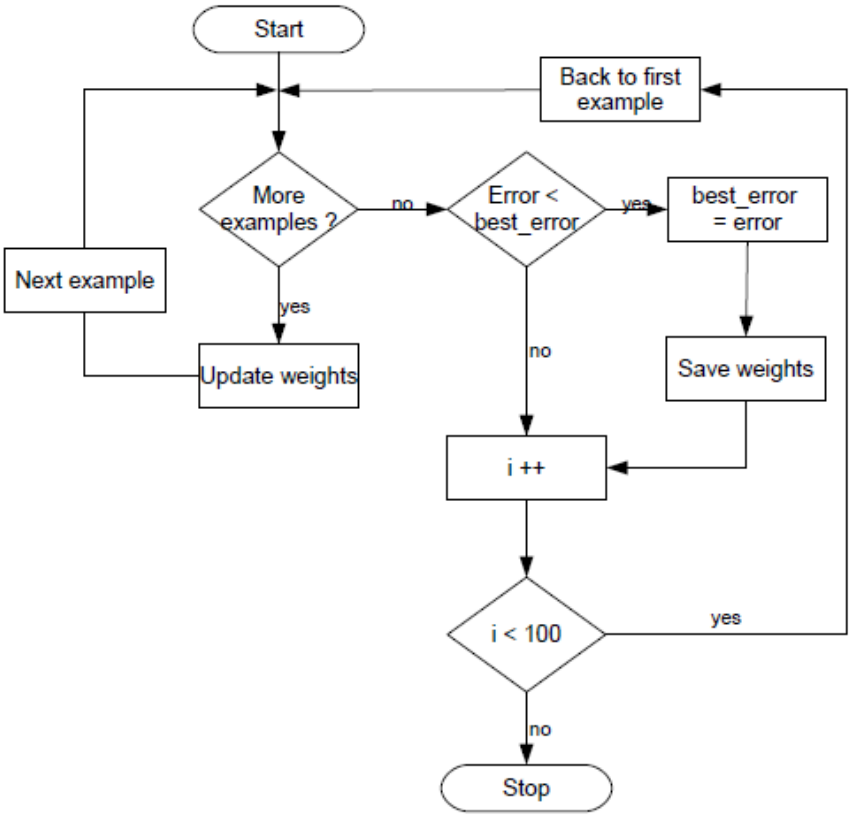

La somme de n nombre



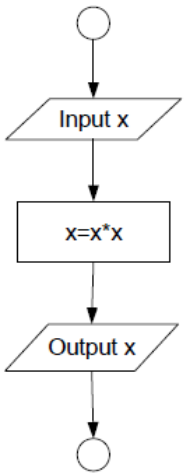

Le tri à bulles

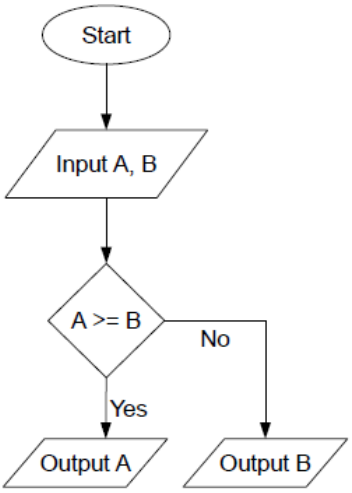


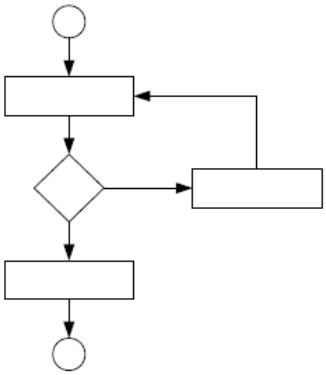
Apprentissage d'un réseau de neurones




Autres



# Annexe C – Résultats détaillés du meilleur reconnaisseur

Nous allons dans cette section présenter quelques résultats supplémentaires obtenus du meilleur reconnaisseur d'expression (Gauss-IroCiel), cf. chapitre 5 (p X). Le tableau suivant résume les taux obtenus sur les trois bases de test.

| Base de test | segRate | recoRate | expRate |
|---|---|---|---|
| RamanReduced_CIEL | 94,3 | 92,1 | 71,3 |
| RamanReduced_Réelle | 89,5 | 84,8 | 49,1 |
| RamanReduced_Wiki_CIEL | 86 | 81,4 | 52,6 |
| Moyenne | 94 | 91,7 | 69,4 |

Les résultats présentés sont des résultats intermédiaires qui nous ont permis de comprendre le comportement du système et découvrir ses points faibles. Ce que nous a amené au meilleur reconnaisseur.

## 1. Taux de reconnaissance par expression

Le taux de reconnaissance d'expression n'est pas suffisant au soit pour détecter les expressions (plus précisément les structures) qui pose le plus des problèmes. Pour cette raison, nous calculons le taux de reconnaissance individuel de chacune des expressions. Le tableau suivant montre les taux obtenus du reconnaisseur « Gauss-IroCiel » sur les base RamanReduced_CIEL et Réelle. Les taux des expressions de deuxième base sont moins significatifs car il n'y a que deux exemples de chacune des expressions. Du même la base RamanReduced_Wiki_CIEL n'est pas outil à cette évaluation car toutes les expressions sont différentes. Dans ce cas, une expression mal reconnue ne signifie pas forcement une difficulté de sa structure. Nous pouvons constater trois catégories des expressions. Celles qui sont reconnues avec un bon taux (supérieur à 70%). Ce taux relativement élevé signifie que la structure de ces expressions est facile à reconnaître et qu'on arrive à surmonter les ambiguïtés par la grammaire et l'analyse structurelle. Les expressions mal reconnues de cette catégorie sont dû aux problèmes de segmentations ou de reconnaissance des symboles. C'est le cas des expressions (1…16, 19, 20, 21,25…30, 33, 34).

La deuxième catégorie (plus petite) comporte des expressions avec plus d'ambiguïtés (expression : 17, 18, 22, 23, 24, 31, 32, 36). La présence de plusieurs symboles qui sont reliés par les relations (gauche/droite, exposant, indice) augmente l'ambiguïté dans l'expression car le nombre des situations valides sera très grand. Une autre raison de la difficulté portée par ces expressions est la présence des symboles de types « fonctions ». Ces symboles varient beaucoup d'un scripteur à un autre. Certains écrivent une fonction (le cosinus par exemple) en cursive avec un seul trait. D'autres découpent les lettres formants de l'expression. Cependant, le taux de reconnaissance de ces expressions et autre de 50%.

Finalement, il y a une seule expression de ce corpus qui n'est pas facilement reconnue (l'expression 35). En plus du problème de présence de deux indices, le point du 'i' perturbe la reconnaissance car il peut être très éloigné ou très près.

Donc, le choix de l'hauteur de ce symbole n'est pas évident. Ce problème peut être résolu en imposant d'un traitement spécifique du cas de 'i'.

| | Expressions | RD_CIEL | RD_Réelle |
|---|---|---|---|
| **1** | a+b+c+d | 88 | 50 |
| **2** | a+\frac{b}{c} +d | 89 | 50 |
| **3** | \frac{a+b}{c+d} | 89 | 50 |
| **4** | \frac{a}{b}+c+d | 87 | 50 |
| **5** | \frac{a}{b+c+d} | 88 | 100 |
| **6** | a+\frac{b+c}{d+e}+x | 79 | 50 |
| **7** | a+bc+d | 76 | 0 |
| **8** | (a+b)(c+d) | 78 | 50 |
| **9** | \frac{x+y^{2}}{k+1} | 80 | 0 |
| **10** | \frac{x+y^{2}}{ k}+1 | 81 | 0 |
| **11** | x+\frac{y^{2}}{k+1} | 80 | 100 |
| **12** | x+\frac{y^{2}}{ k}+1 | 80 | 50 |
| **13** | x+y^{\frac{2}{ k+1}} | 82 | 50 |
| **14** | (x^{2})^{y} = x^{2y} | 70 | 50 |
| **15** | 1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\frac{x}{1+\frac{x}{1+ \cdots }}}}} | 71 | 50 |
| **16** | (a+b)^{3}=a^{3}+3a^{2}b+3ab^{2}+b^{3} | 76 | 0 |
| **17** | a^{3}+b^{3}=(a+b)(a^{2}-ab+b^{2}) | 53 | 50 |
| **18** | x= \frac{-b \pm \sqrt{b^{2}-4ac}}{2a} | 44 | 0 |
| **19** | \frac{1+\sqrt{5}}{2}=\phi | 76 | 100 |
| **20** | \frac{\sqrt{\pi}}{2} \neq \sqrt{\frac{\pi}{2}} | 77 | 100 |
| **21** | \sin^{2}x+\cos^{2}x=1 | 69 | 0 |
| **22** | \sin x^{2} + \cos x^{2} \neq 1 | 62 | 0 |
| **23** | \sin (a+b) = \sin a \cos b + \cos a \sin b | 57 | 50 |
| **24** | \cos (x+y)=\cos x \cos y - \sin x \sin y | 58 | 0 |
| **25** | \sin 2x = 2 \sin x \cos x | 72 | 0 |
| **26** | \cos 2x = \cos^{2} x -\sin^{2} x | 70 | 50 |
| **27** | \log^{2}x\neq2\log x | 75 | 100 |
| **28** | \log x^{2}=2\log x | 80 | 100 |
| **29** | \frac{\log x}{\log a} = \log_{a} x | 76 | 0 |
| **30** | 1+x+\frac{x^{2}}{2}+\frac{x^{3}}{3}+\cdots +\frac{x^{n}}{n}+\cdots | 71 | 0 |
| **31** | x - \frac{x^{2}}{2} +\frac{x^{3}}{3}- \frac{x^{4}}{4}+\frac{x^{5}}{5} \pm \cdots = \log(1+x) | 51 | 0 |
| **32** | \gamma = 1+\frac{1}{2}+\frac{1}{3} +\frac{1}{4} +\cdots +\frac{1}{n}-\log n | 54 | 0 |
| **33** | \int\frac{dx}{x} =\log x | 78 | 50 |
| **34** | \sum_{i=1}^{n} a_{i} =1 | 81 | 50 |
| **35** | \sum_{i=1}^{n} a_{i} + b_{i} = 1 | 5 | 0 |
| **36** | \cosh^{2}x -\sinh^{2} x = 1 | 53 | 0 |

## 2. Taux de reconnaissance d'expressions en considérant les meilleurs cinq résultats

Considérer les meilleurs 5 candidats montre qu'on peut surmonter quelques erreurs de reconnaissance où le taux moyenne de reconnaissance d'expressions s'augmente de 69.4% à 75.6%. Cela est surtout bénéfique avec une interface interactive de saisi. Par exemple on peut proposer plusieurs solutions à l'utilisateur en cas de doute.

| | RD_CIEL | RD_Réelle | RD_Wiki_CIEL | Moyenne |
|---|---|---|---|---|
| **Top1** | 71 | 38,6 | 52,6 | 69,4 |
| **Top2** | 75,3 | 50 | 58,8 | 74 |
| **Top3** | 76 | 50 | 62,1 | 74,8 |
| **Top4** | 76,5 | 50 | 63,5 | 75,3 |
| **Top5** | 76,7 | 50 | 64,9 | 75,6 |

## 3. Taux de reconnaissance par symbole

Ce tableau montre le taux de reconnaissance individuel de chaque symbole. La plupart des symboles sont reconnus avec un bon taux (supérieur à 90%). Nous pouvons conclure que la faiblesse face aux expressions ambiguës vient de l'analyse structurelle et que le classifieur est bien appris grâce à l'apprentissage global.

| Symbole | RD_CIEL | RD_Réelle | RD_Wiki_CIEL |
|---|---|---|---|
| **1** | 92,65 | 88,37 | 88,32 |
| **2** | 95,78 | 77,46 | 87,10 |
| **3** | 97,58 | 86,96 | 88,24 |
| **4** | 93,50 | 71,43 | 92,31 |
| **5** | 91,00 | 100,00 | 92,31 |
| **a** | 93,00 | 87,76 | 88,61 |
| **b** | 94,50 | 88,37 | 84,85 |
| **c** | 96,67 | 88,89 | 91,67 |
| **d** | 94,78 | 88,89 | 62,50 |
| **e** | 91,00 | 50,00 | 78,95 |
| **i** | 89,80 | 100,00 | 78,85 |
| **k** | 91,60 | 80,00 | 58,82 |
| **n** | 95,67 | 55,56 | 83,05 |
| **x** | 90,67 | 76,09 | 85,54 |
| **y** | 94,60 | 75,00 | 95,45 |
| **+** | 92,11 | 81,03 | 79,82 |
| **-** | 95,76 | 85,33 | 95,39 |
| **=** | 93,10 | 89,19 | 87,00 |
| **pm** | 88,50 | 75,00 | NA |
| **neq** | 90,33 | 83,33 | 100,00 |
| **gamma** | 86,00 | 100,00 | 100,00 |
| **pi** | 85,50 | 100,00 | 88,89 |

| | | | |
|---|---|---|---|
| **phi** | 87,00 | 100,00 | NA |
| **sum** | 94,50 | 100,00 | 66,67 |
| **int** | 95,00 | 100,00 | NA |
| **sqrt** | 84,75 | 87,50 | 71,05 |
| **(** | 85,33 | 72,22 | 92,54 |
| **)** | 85,56 | 72,22 | 92,54 |
| **cdots** | 79,00 | 77,78 | 100,00 |
| **cos** | 91,70 | 65,00 | 100,00 |
| **sin** | 77,80 | 70,00 | 63,64 |
| **sinh** | 67,00 | 50,00 | 50,00 |
| **cosh** | 81,00 | 100,00 | 50,00 |
| **log** | 86,50 | 73,68 | 100,00 |

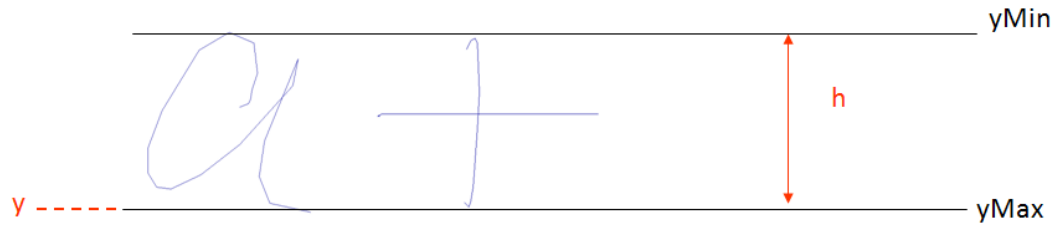
# Annexe D – Informations supplémentaires pour l'analyse contextuelle

## 1. Les informations spatiales selon le type du symbole

Pour tous les types des symboles, les valeurs de taille *h* et du positionnement *y* sont calculées en fonction de *yMin*, *yMax* du symbole.

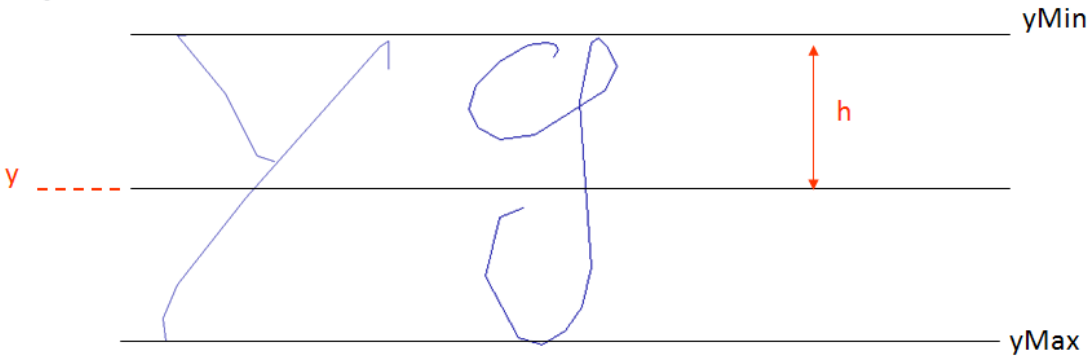
Les symboles réguliers

$a,c,e,m,n,o,r,s,u,v,w,x,+,\leq,\neq,cos,\pm,\div,\gamma,\pi,\vartheta,..etc$



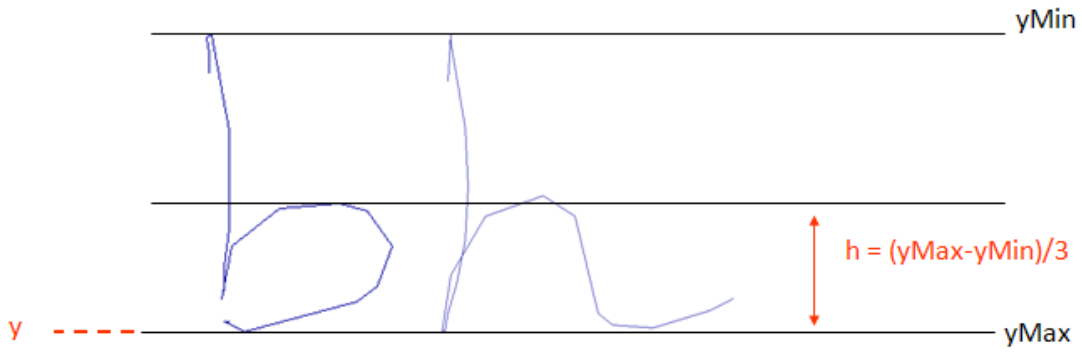

Les symboles descendants

$g,j,p,q,y,z$

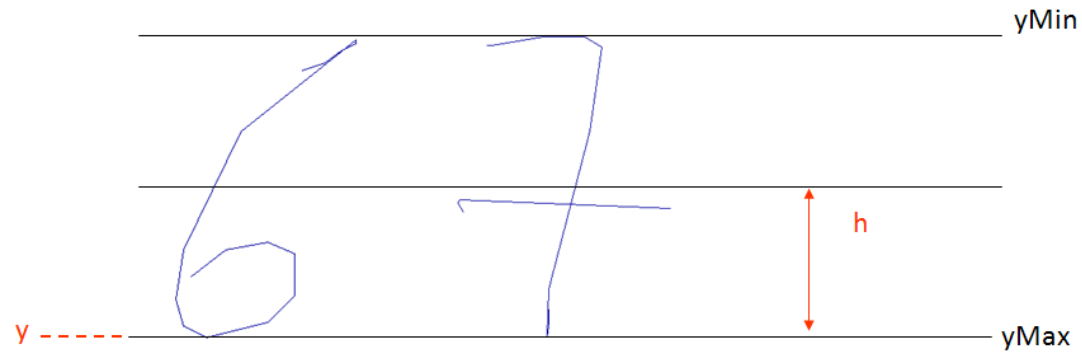



Les symboles ascendants

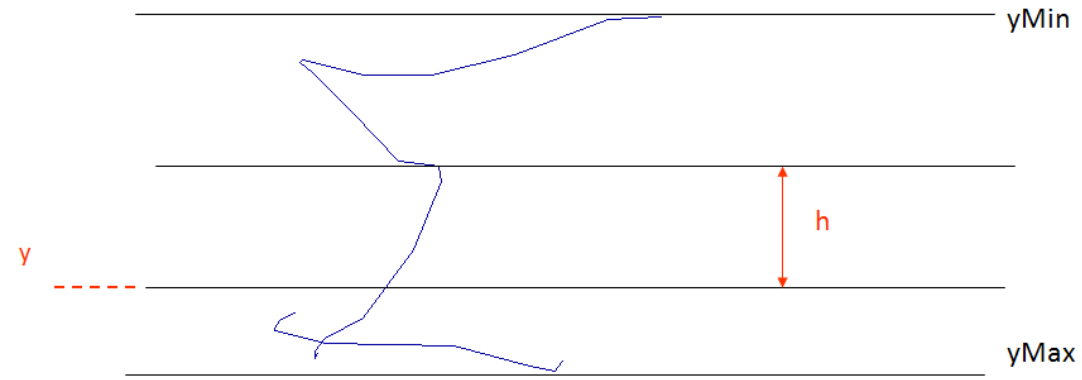
$b,d,f,h,k,l,t,lim$



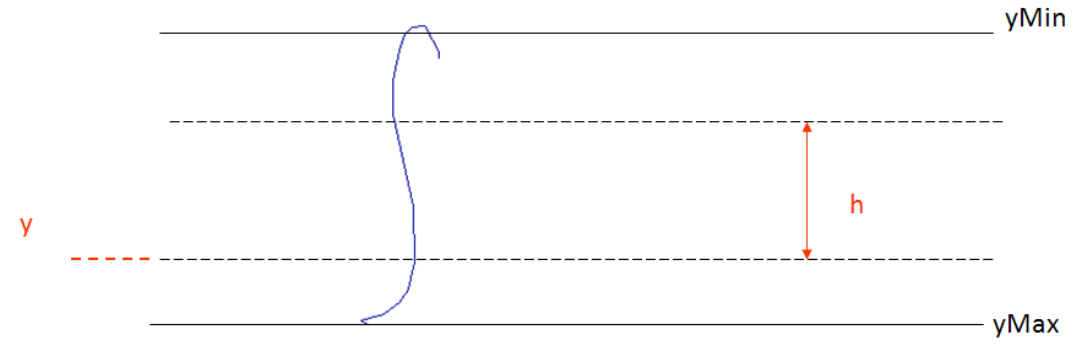

Les chiffres



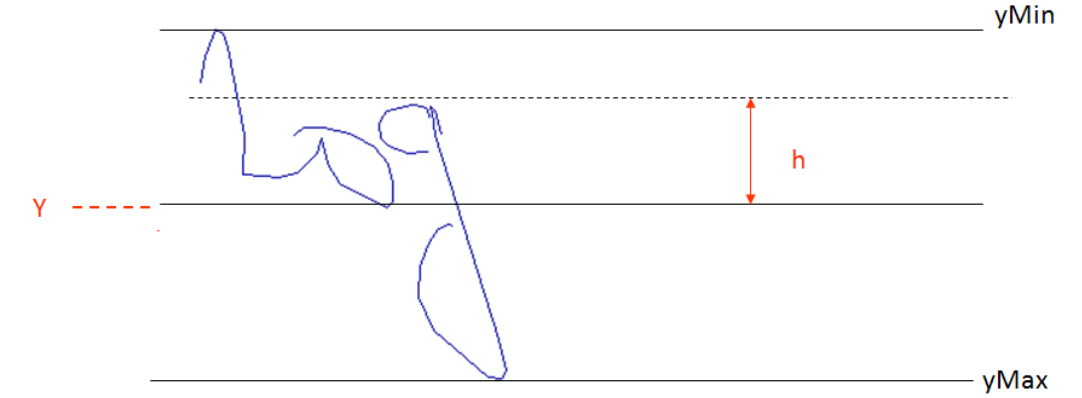

La somme




L'intégrale

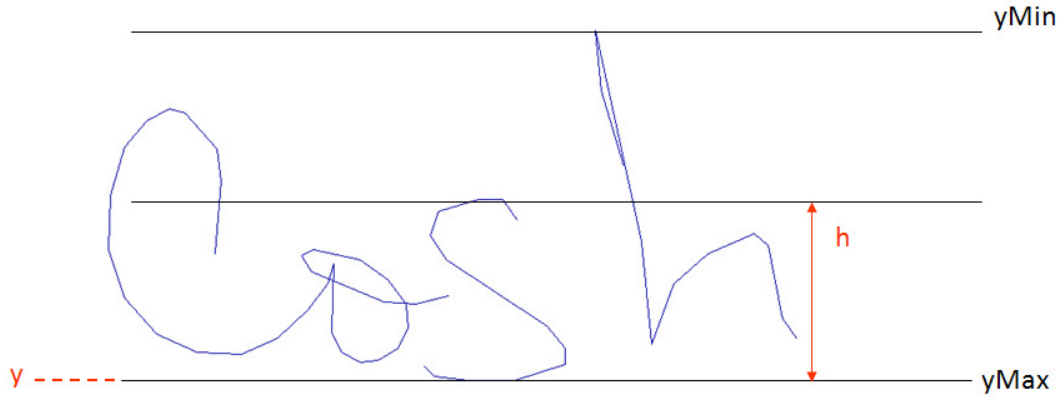



Log



Autres

$sin, sinh, cosh$




## 2. La grammaire

Ci-après la grammaire utilisée dans le modèle de langage du système pour la reconnaissance d'expressions du corpus RamanReduced.

HP : Horizontal Pair Rule, SupR : Superscript Rule, SubR : Subscript Rule, IR : Integral Rule, SqrtR : Sqrt Rule, OR : Operation Rule, FR : Filter Rule, IDR : Identity Rule, FrR : Fractionbar Rule, SR : Sum Rule, PR : Paranthesis Rule, SigSR : SigmaSum Rule

FRs :
Digit ← 0...9 , Letter ←a..z | A..Z , dots ←\\cdots, Other ← | \\gamma | \\theta | \\pi | \\phi | \\infty , Symbol ← Digit | Letter | Other
Op ← + | - | \\pm, Fractionbar ← - , Eq ← = | \\neq | \\leq, Sqrt ← \\sqrt
Integral ← \\int, Sigma ← \\sum, OpenP ← (, CloseP ← )
Func ← \\sin \\cos \\sinh \\cosh \\log

Sum grammar :
Sum ← Op notSum *(HP)* | Exp Op NotSum *(OR) )* | Exp Op dots *(OR)*

NotSumGrammar :
NotSum ← NotSum NotSum *(HP)* | Symbol *(IDR)*

Functions grammar :
Function ← Func Exp *(SupR)* | Func symbol *(subR)* | Func *(IDR)*
NotSum ← Function notSum *(HP)*

Sigma grammar :
Assignement ← Symbol Eq Exp *(OR)* | Exp *(IDR)*
SigmaSum ← Exp Sigma Assignement *(SR)* | Sigma *(IDR)*
NotSum ← SigmaSum NotSum *(SigSR)*

Paranthesis grammar :

ParExp ← OpenP Exp CloseP *(PR)*
NotSum ← ParExp *(IDR)*

Fractionbar grammar :
NotSum ← Exp FractionBar Exp *(FrR)*

Sqrt grammar :
NotSum ← Sqrt Exp *(SqrtR)*

Integral grammar :
NotSum ← Integral Exp NotSum Exp *(IR)*

Sub/Supscript grammar :
NotSum ← NotExp Exp *(SupR)* | Symbol Symbol *(SubR)*
NotExp ← Symbol | Func | ParExp *(IDR)*

Grammar :
Exp ← NotSum | Sum *(IDR)*
Formula ← Exp Eq Exp *(OR)* | Exp *(IDR)*
S ← Formula

# BIBLIOGRAPHIE

## Références de l'auteur


a1. Ahmad-Montaser Awal, Guihuan Feng, Harold Mouchère et Christian Viard-Gaudin. **First Experiments on a new Online Handwritten Flowchart Database.** *Document Recognition and Retrieval XVIII, San Francisco, jan* 2011.

a2. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **Improving online handwritten mathematical expressions recognition with contextual modeling.** *International Conference on Frontiers in Handwriting Recognition, Calcutta,* Nov 2010.

a3. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **The Problem of Handwritten Mathematical Expression Recognition Evaluation**. *International Conference on Frontiers in Handwriting Recognition, Calcutta,* Nov 2010**.**

a4. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **Apprentissage de relations spatiales pour la reconnaissance d'expressions mathématiques manuscrites en-lignes.** *Colloque International Francophone sur l'Ecrit et le Document, Sousse,* 2010, 265-280.

a5. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **A hybrid classifier for handwritten mathematical expression recognition.** *Document Recognition and Retrieval XVII, San Jose,* 2010, 1-10.

a6. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **Un classifieur hybride pour la reconnaissance d'expressions mathématiques manuscrites en-lignes**. *Conférence Reconnaissance des Formes et Intelligence Artificielle - RFIA, Caen,* 2010, 487-494.

a7. Ahmad-Montaser Awal, Harold Mouchère et Christian Viard-Gaudin. **Towards handwritten mathematical expression recognition**. *Tenth International Conference on Document Analysis and Recognition, Barcelone,* 2009, 1046-1050.

a8. Ahmad-Montaser Awal, Romain Cousseau et Christian Viard-Gaudin. **Convertisseur d'équations LATEX2Ink**. *Colloque International Francophone sur l'Ecrit et le Document, Rouen,* 2008, 193-194.

a9. Ahmad-montaser Awal et Christian Viard-Gaudin**. Towards Handwritten Mathematical Expressions Recognition**. *First gCEO Keio-Centrale Nantes workshop, octobre* 2008.

# Références


[1] Guihuan Feng, "Sketch understanding techniques for informal human-computer interaction," Nanjing University, Thèse de doctorat 2009.

[2] Réjean Plamondon and Sargur N. Srihari, "On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 63-84, 2000.

[3] Kam-Fai Chan and Dit-Yan Yeung, "Mathematical expression recognition: A survey," *International journal on Document anlysis and recognition*, vol. 3, pp. 3-15, 2000.

[4] ZM. Yuan, H. Pan, and L. Zhang, "A novel pen-based flowchart recognition system for programming teaching," *Lecture Notes in Computer Science*, vol. 5328, pp. 55-64, 2008.

[5] Gaurangi Tilak and Krithika Ananthakrishnan, "SketchUML – Sketch based approach to Class Diagrams," in *IUI Workshop on Sketch Recognition*, 2009.

[6] Bertrand Coüasnon, "DMOS: a generic document recognition method, application to an automatic generator of musical scores, mathematical formulae and table structures recognition systems," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 215-220.

[7] Xin Wang, Guangshun Shi, and Jufeng Yang, "The Understanding and Structure Analyzing for Online Handwritten Chemical Formulas," in *Tenth International Conference on Document Analysis and Recognition*, 2009, pp. 1056-1061.

[8] Mariusz Szwoch, "Guido: a musical score recognition system," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 809-813.

[9] Adrien Delaye, Eric Anquetil, and Sébastien Macé, "Explicit fuzzy modeling of shapes and positioning for handwritten Chinese character recognition," in *Tenth International Conference on Document Analysis and Recognition*, 2009, pp. 1121-1125.

[10] Mohamed Cheriet, Mounim A. El-Yacoubi, Hiromichi Fujisawa, Daniel P. Lopresti, and Guy Lorette, "Handwriting recognition research: Twenty years of achievement. and beyond," *Pattern Recognition*, vol. 42, pp. 3131-3135, 2009.

[11] Christian Viard-Gaudin, Pierre-Michel Lallican, and Stefan Knerr, "Recognition-directed recovering of temporal information from handwriting images," *Pattern Recognition Letters*, vol. 26, pp. 2537-2548, 2005.

[12] Pierre-Michel Lallican, "Reconnaissance de l'écriture manuscrite hors-ligne : utilisation de la chronologie restaurée du tracé," Université de Nantes, Thèse de doctorat 1999.

[13] Laëtitia Rousseau, "Reconnaissance d'écriture manuscrite hors-ligne par reconstruction de l'ordre du tracé en vue de l'indexation de document d'archives," Institut National des Sciences Appliquées de Rennes, Thèse de doctorat 2007.

[14] Seong-Whan Lee, , Henry S. Baird, Horst Bunke, and Kazuhiko Yamamoto, Eds.: Springer-Verlag, 1992, ch. Recognizing hand-drawn electrical circuit symbols with attributed graph matching, pp. 340-357.

[15] Taik-Heon Rhee and Jin-Hyung Kim, "Efficient search strategy in structural analysis for handwritten mathematical expression recognition," *Pattern Recognition*, vol. 42, pp.

3192-3201, 2009.

[16] Guihuan Feng, Christian Viard-Gaudin, and Zhengxing Sun, "On-line hand-drawn electric circuit diagram recognition using 2D dynamic programming," *Pattern Recognition*, vol. 42, pp. 3215-3223, 2009.

[17] Daniel Prusa, "Two-dimensional Context-free Grammars," in *ITAT*, 2001, pp. 27-40.

[18] Jesse F. Hull, "Recognition of Mathematics Using a Tow-Dimensional Trainable Context-free Grammar," Massachuestts Institute Of Technology, Thèse de Master 1996.

[19] Daniel Prusa and Vaclav Hlavac, "Mathematical formulae recognition using 2D grammars," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 849-853.

[20] Yu-Sheng Guo, Lei Huang, Chang-Ping Liu, and Xin Jiang, "An automatic mathematical expression understanding system," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 719-723.

[21] Erik G. Miller and Paul A. Viola, "Ambiguity and Constraint in Mathematical Expression Recognition," in *The 15th national conference on artificial intelligence*, 1998, pp. 784-791.

[22] William P. Berlinghoff and Fernando Quadros Gouvêa, *Math through the ages: a gentle history for teachers and others*, 0, Ed.: Oxton house publishers & The mathematical association of America, 2004.

[23] Ryoji Fukuda, Sou I, Fumikazu Tamari, Xie Ming, and Masakazu Suzuki, "A technique of mathematical expression structure analysis for the handwriting input system," in *Fifth International Conference on Document Analysis and Recognition*, 1999, pp. 131-134.

[24] Ryo Yamamoto, Shinji Sako, Takuya Nishimoto, and Shigeki Sagayama, "On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-Free Grammar," in *tenth International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 249-254.

[25] William A. Martin, "Computer input/output of mathematical expressions," in *second ACM symposium on Symbolic and algebraic manipulation*, 1971, pp. 78-89.

[26] Sébastien Macé, "Composition manuscrite interactive et interprétation à la volée de documents structurés en-ligne," INSA de Rennes, Thèse de doctorat 2008.

[27] Eric Anquetil, "Modélisation et reconnaissance par la logique floue : application à la lecture automatique de l'écriture manuscrite omni-scripteur," Université de Rennes I, Thèse de doctorat 1997.

[28] Eric Anquetil and Hélène Bouchereau, "Integration of an on-line handwriting recognition system in a smart phone device," in *Sixteenth International Conference on Pattern Recognition*, vol. 3, 2002, pp. 192–195.

[29] Jamie Anstice, Tim Bell, Andy Cockburn, and Martin Setchell, "The design of a penbased musical input system," in *Sixth Australian Conference on Computer-Human Interaction*, 1996, pp. 260–267.

[30] Tracy Hammond and Randall Davis, "Tahuti : a geometrical sketch recognition system for uml class diagrams," in *the AAAI Spring Symposium on Sketch Understanding*, 2002,

pp. 59-68.

[31] Ernesto Tapia and Raul Rojas, "Recognition of on-line handwritten mathematical expressions in the E-Chalk system - An extension," in *Eighth International Conference on Document Analysis and Recognition*, vol. 2, 2005, pp. 1206-1210.

[32] Joseph J. LaViola and Robert C. Zeleznik, "MathPad2: A System for the Creation and Exploration of Mathematical Sketches," *ACM Transactions on Graphics*, vol. 23, pp. 432-440, 2004.

[33] Kam-Fai Chan and Dit-Yan Yeung, "PenCalc: a novel application of on-line mathematical expression recognition technology," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 775-778.

[34] J. Hartmann et al., "The "Optical Formula Recognition" System for Handprinted Input," *Symbolic Computation*, vol. 11, pp. 1-8, 1995.

[35] Robert H. Anderson, "Syntax-directed recognition of handprinted two-dimensional mathematics," in *Interactive Systems for Experimental Applied Mathematics*, 1968, pp. 436-459.

[36] S.K. Chang, "A method for the structural analysis of 2-D mathematical expressions," *Information Sciences*, vol. 2, pp. 253-272, 1970.

[37] Abdelwahed Belaid and Jean Paul Haton, "A syntactic approach for handwritten mathematical formulae recognition," *Pattern Analysis and Machine Intelligence*, vol. 6, pp. 105-111, 1984.

[38] Ernesto Tapia and Raul Rojas, "A Survey on Recognition of on Line Handwritten Mathematical Notation," Freie Universit¨at Berlin, Institut f¨ur Informatik, Germany, 2007.

[39] Abdul Rahim Ahmad, Christian Viard-Gaudin, and Marzuki Khalid, "Lexicon-based Word Recognition Using Support Vector Machine and Hidden Markov Model," in *Tenth International Conference on Document Analysis and Recognition*, 2009, pp. 161-166.

[40] Claude Faure and Zi Xiong Wang, , Réjean Plamondon and C. Graham Leedham, Eds.: World Scientific, 1990, ch. Automatic perception of the structure of handwritten mathematical expressions, pp. 337-361.

[41] Masayuki Okamoto and Bin Miao, "Recognition of Mathematical expressions by using the layout structures of symbols," in *First International Conference on Document Analysis and Recognition*, vol. 1, 1991, pp. 242-250.

[42] Jaekyu Ha, Robert M. Haralick, and Ihsin T. Philips, "Understanding mathematical expressions from document images," in *Third International Conference on Document Analysis and Recognition*, vol. 2, 1995, pp. 956-959.

[43] Hsi-Jian Lee and Min-Chou Lee, "Understanding Mathematical expressions in a printed document," in *Second International Conference on Document Analysis and Recognition*, 1993, pp. 502-505.

[44] Hsi-Jian Lee and Min-Chou Lee, "Understanding mathematical expressions using procedure-oriented transformation," *Pattern Recognition*, vol. 27, pp. 447-457, 1994.

[45] Utpal Garain and B.B. Chaudhuri, "Segmentation of touching symbols for OCR of printed mathematical expressions: An approach based on multifactorial analysis," in

*Eighth International Conference on Document Analysis and Recognition*, vol. 1, 2005, pp. 177-181.

[46] Akihiro Nomura, Kazuyuki Michishita, Seiichi Uchida, and Masakazu Suzuki, "Detection and segmentation of touching characters in mathematical expressions," in *Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 126-130.

[47] Masayuki Okamoto, Syougo Sakaguchi, and Tadashi Suzuki, "Segmentation of Touching Characters in Formulas," in *Third IAPR Workshop on Document Analysis Systems*, 1998, pp. 283-289.

[48] James A. Landay and Brad A. Myers, "Interactive sketching for the early stages of user interface design," in *the SIGCHI conference on Human factors in computing systems*, 1995, pp. 43–50.

[49] Beryl Plimmer and Mark Apperley, "Software to sketch interface designs," in *Ninth International Conference on Human-Computer Interaction*, 2003, pp. 73–80.

[50] Christian Heide Damm, Klaus Marius Hansen, and Michael Thomsen, "Tool support for cooperative object-oriented design : gesture based modelling on an electronic whiteboard," in *SIGCHI conference on Human factors in computing systems*, 2000, pp. 518–525.

[51] Sébastien Macé, Eric Anquetil, Elodie Garrivier, and Bruno Bossis, "A pen-based musical score editor," in *International Computer Music Conference*, 2005, pp. 415–418.

[52] Yannis A. Dimitriadis, Jaun Lopez Coronado, and Cristina de la, "A new interactive mathematical editor, using on-line handwritten symbol recognition, and error detection-correction with an attribute grammar," in *First International Conference on Document Analysis and Recognition*, 1991, pp. 885-893.

[53] Ernesto Tapia and Raul Rojas, "Recognition of on-line handwritten mathematical formulas in the E-Chalk system," in *Seventh International Conference on Document Analysis and Recognition*, vol. 2, 2003, pp. 980-984.

[54] Shi-Zhong Liao, Wen-Gang Liu, and Wei Guo, "Composite sketch shape recognition based on dagsvm and decision tree," in *Fifth International Conference on Machine Learning and Cybernetics*, 2006, pp. 3254–3259.

[55] Ernesto Tapia, "Understaning Mathematics: A system for the recognition of on-line handwritten mathematical expressions," Fachbereich Mathematik und Informatik der Freien Universit at Berlin, Thèse de doctorat 2004.

[56] Stefan Lehmberg, Hans-Jürgen Winkler, and Manfred Lang, "A Soft-decision approach for symbol segmentation within handwritten mathematical expressions," in *International Conference on Acoustics, Speech, and Signal Processing*, 1996, pp. 3434-3437.

[57] Leslie Gennaria, Levent Burak Karaa, Thomas F. Stahovich, and Kenji Shimada, "Combining geometry and domain knowledge to interpret hand-drawn diagrams," *Computer and Graphics*, vol. 29, pp. 547-562, 2005.

[58] Y. Nakayama, "A prototype pen-input mathematical formula editor," in *ED-MEDIA 93-World Conference on Educational Multimedia and Hypermedia*, 1993, pp. 400-407.

[59] Kam-Fai Chan and Dit-Yan Yeung, "An efficient syntactic approach to structural analysis

of on-line handwritten mathematical expressions," *Pattern recognition*, vol. 33, pp. 375-384, 2000.

[60] Xiaogang Xu, Zhengxing Sun, Binbin Peng, Wiangyu Jin, and Wenyin Liu.s., "An online composite graphics recognition approach based on matching of spatial relation graph," *International Journal on Document Analysis and Recognition*, vol. 7, pp. 44–55, 2004.

[61] Steve Smithies, Kevin Novins, and James Arvo, "A Handwriting-Based Equation Editor," in *Graphics Interface '99*, 1999, pp. 84-91.

[62] Corinna Cortes and Vladimir Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, pp. 273-297, 1995.

[63] Birendra Keshari and Stephen M. Watt, "Hybrid mathematical symbol recognition using support vector machines," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 859-863.

[64] Hidetoshi Miyao and Minoru Maruyama, "An online handwritten music symbol recognition system," *International Journal on Document Analysis and Recognition*, vol. 9, pp. 49–58, 2007.

[65] E. Anquetil and G. Lorette, "On-Line Handwriting Character Recognition System Based on Hierarchical Qualitative Fuzzy Modeling," in *Fifth International Workshop on Frontiers in Handwriting Recognition*, 1996, pp. 47-52.

[66] Sébastien Macé and Eric Anquetil, "Eager Interpretation of On-Line Hand-Drawn Structured Documents: The DALI Methodology," *Pattern Recognition*, vol. 42, pp. 3202-3214, 2009.

[67] Ray Geneo, John A. Fitzgerald, and Tahar Kechadi, "A Purely Online Approach to Mathematical Expression Recognition," in *International Workshop on Frontiers in Handwriting Recognition*, 2006, pp. 255-260.

[68] John A. Fitzgerald, Franz Geiselbrechtinger, and Tahar Kechadi, "Mathpad : A fuzzy logic-based recognition system for handwritten mathematics," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 694-698.

[69] J.L. McClelland and D.E. Rumelhart,.: Cambridge, MIT Press, 1986, vol. é.

[70] W.S. McCulloch and W. A. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys*, vol. 5, pp. 115-133, 1943.

[71] Susan E. George, "Online pen-based recognition of music notation with artificial neural networks," *Computer Music Journal*, vol. 27, pp. 70–79, 2003.

[72] Yannis A. Dimitriadis and Juan Lopez Coronado, "Towards an ART based mathematical editor that uses online handwritten symbol recognition," *Pattern Recognition*, vol. 28, pp. 807-822, 1995.

[73] R. Marzinkewitsch, "Operating computer algebra systems by handprinted document," in *International Symposium on Symbolic and Algebraic Computation*, 1991, pp. 411-413.

[74] Emilie Poisson, "Architecture et apprentissage d'un système hybride neuro-markovien pour la reconnaissance de l'écriture manuscrite en-ligne," Ecole polytechnique de l'université de Nantes, Thèse de doctorat 2005.

[75] Lawrence R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," in *Proceedings of the IEEE*, 1989, pp. 257-286.

[76] Andreas Kosmala, Gerhard Rigoll, Stéphane Lavirotte, and Loïc Pottier, "On-Line Handwritten Formula Recognition using Hidden Markov Models and Context Dependent Graph Grammars," in *Fifth International Conference on Document Analysis and Recognition*, 1999.

[77] Jean-Philippe Valois, Myriam Côté, and Mohamed Cheriet, "Online recognition of sketched electrical diagrams," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 460-464.

[78] Kam-Fai Chan and Dit-Yan Yeung, "Error detection, error correction and performance evaluation in on-line mathematical expression recognition," *Pattern Recognition*, vol. 34, pp. 1671-1684, 2001.

[79] Hsi-Jian Lee and Jiumn-Shine Wang, "Design of mathematical expression recognition system," in *Third International Conference on Document Analysis and Recognition*, vol. 2, 1995, pp. 1084-1087.

[80] Kang Kim, Taik Heon Rhee, Jae Seung Lee, and Jin Hyung Kim, "Utilizing Consistency Context for Handwritten Mathematical Expression Recognition," in *Tenth International Conference on Document Analysis and Recognition*, 2009, pp. 1051-1056.

[81] Hashim M. Twaakyondo and Masayuki Okamoto, "Structure analysis and recognition of mathematical expressions," in *Third International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 430-437.

[82] Hsi-Jian Lee and Jiumn-Shine Wang, "Design of a mathematical expression understanding system," *Pattern RecognitionLetters*, vol. 18, pp. 289-298, 1997.

[83] Adrien Delaye and Harold Mouchère, "Vers une approche générique pour la reconnaissance de formes manuscrites structurées," in *Colloque International Francophone sur l'Ecrit et le Document*, 2010.

[84] Richard Zanibbi, Dorothea Blostein, and James R. Cordy, "Baseline stucture analysis of handwritten mathematics notation," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 769-773.

[85] Richard Zanibbi and Dorothea Blostein, "Recognizing Mathematical Expressions Using Tree Transformation," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1455-1467, 2002.

[86] Joydip Mitra, Utpal Garain, B.B. Chaudhuri, Kumar Swamy H.V., and Tamaltaru Pal, "Automatic understanding of structures in printed mathematical expressions," in *Seventh International Conference on Document Analysis and Recognition*, vol. 1, 2003, pp. 540-544.

[87] Yuko Eto and Masakazu Suzuki, "Mathematical formula recognition using virtual link network," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 762-767.

[88] Walaa Aly, Seiichi Uchida, Akio Fujiyoshi, and Masakazu Suzuki, "Statistical classification of spatial relationships among mathematical symbols," in *Tenth International Conference on Document Analysis and Recognition*, 2009, pp. 1350-1355.

[89] Stéphane Lavirotte, "Reconnaissance structurelle de formules mathématiques typographiées et manuscrites," UNIVERSIT´EDE NICE – SOPHIA ANTIPOLIS ´Ecole Doctorale des Sciences et Technologies de l'Information et de la Communication, Thèse de doctorat 2000.

[90] Stéphane Lavirotte and Loïc Pottier, "Mathematical Formula Recognition Using Graph Grammar," in *In Proceedings of the SPIE*, 1998, pp. 44-52, Printed version is available.

[91] Ling Zhang, Dorothea Blostein, and Richard Zanibbi, "Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions," in *Eighth International Conference on Document Analysis and Recognition*, vol. 2, 2005, pp. 972-976.

[92] John A. Fitzgerald, Franz Geiselbrechtinger, and Tahar Kechadi, "Structural Analysis of Handwritten Mathematical Expressions Through Fuzzy parsing," in *The International Conference on Advances in Computer Science and Technology*, 2006, pp. 151-156.

[93] Daniel Prusa and Vaclav Hlavac, "2D Context-Free Grammars: Mathematical Formulae Recognition," in *The Prague Stringology Conference*, 2006, pp. 77-89.

[94] Utpal Garain and B. Chaudhuri, "Recognition of Online Handwritten Mathematical Expressions," *Transactions on Systems, Man and Cybernetics*, vol. 34, pp. 2366-2376, 2004.

[95] Horst Bunke, "Attributed programmed graph grammars and their application to schematic diagram interpretation," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 574–582, 1982.

[96] Ann Grbavec and Dorothea Blostein, "Mathematics recognition using graph rewriting," in *Third International Conference on Document Analysis and Recognition*, vol. 1, 1995, pp. 417-421.

[97] Yu Shi, Hai Yang Li, and Franck K. Soong, "A unified framwork for symbol segmentation and recognition of handwritten mathematical expressions," in *Ninth International Conference on Document Analysis and Recognition*, 2007, pp. 854-858.

[98] Yusuke Takiguchi and Minoru Okada, "A fundamental study of output translation from layout recognition and semantic understanding system for mathematical formulae," in *Eighth International Conference on Document Analysis and Recognition*, vol. 2, 2005, pp. 745-749.

[99] Masayuki Okamoto, Hiroki Imai, and Kazuhido Takagi, "Prerformance evaluation of a robust method for mathematical expression recognition," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 121-128.

[100] Ihsen T. Phillips, "Methodologies for Using UW Databases for OCR and Image Understanding Systems," in *Document Recognition V*, 1998, pp. 112-127.

[101] Utpal Garain and B.B. Chaudhuri, "A corpus for OCR research on mathematical expressions," *International Journal on Document Analysis and Recognition*, vol. 7, pp. 241-259, 2005.

[102] T.V. Raman, "Audio system for technical readings," Cornell University, Thèse de doctorat 1994.

[103] Utpal Garain and B.B. Chaudhuri, "On machine understanding of online handwritten

mathematical expressions," in *Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 349-353.

[104] Scott MacLean, George Labahn, Edward Lank, Mirette Marzouk, and David Tausky, "Grammar-Based techniques for creating ground-truthed sketch corpora," *International Journal of Document Analysis and Recognition*, vol. Onlinie, p. 1, 2010.

[105] Thomas J. Pennello and Frank DeRemer, "Efficient computation of LALR(1) look-ahead sets," *ACM SIGPLAN Notices*, vol. 39, pp. 14-27, 2004.

[106] Utpal Garain, "Automatic Recognition Of Printed and Handwritten Mathematical Expression," THE INDIAN STATISTICAL INSTITUTE, Thèse de doctorat 2005.

[107] Isabelle Guyon, Lambert Schomaker, Réjean Plamondon, Mark Liberman, and Stan Janet, "UNIPEN Project of On-Line Data Exchange and Recognizer Benchmarks," in *Twelvth International Conference on Pattern Recognition*, 1995, pp. 29-33.

[108] Christian Viard-Gaudin, Pierre-Michel Lallican, S. Knerr, and Philippe Binter, "The IRESTE On/Off (IRONOFF) dual handwriting database," in *Fifth International Conference on Document Analysis and Recognition*, 1999, pp. 455-458.

[109] Kenichi Toyozumi, Takahiro Suzuki, Kensaku Mori, and Yasuhito Suenaga, "A system for real-time recognition of handwritten mathematical formulas," in *Sixth International Conference on Document Analysis and Recognition*, 2001, pp. 1059-1063.

[110] Philip Bille, "A survey on tree edit distance and related problems," *Theory Computrt Science*, vol. 337, pp. 217-239, 2005.

[111] Kunal Sain, Abhishek Dasgupta, and Utpal Garain, "EMERS: a tree matching–based performance evaluation of mathematical expression recognition systems," *International Journal of Document Analysis and Recognition*, vol. Online, pp. 1-11, 2010.

[112] Mohamed Cheriet, Nawwaf Kharma, Cheng-Lin Liu, and Ching Suen, *Character Recognition Systems, a Textbook for Students and Practitioners*, 0, Ed.: John Wiley & Sons, 2004.

[113] J.S. Bridle, , D.S. Touretzky, Ed.: 0, 1990, ch. Advances in Neural Information Processing Systems, pp. 211-217.

[114] Y. LeCun, *Modeles connexionnistes de l'apprentissage (connectionist learning models)*, 0, Ed.: 0, 1987.

[115] Fabrice Rossi, "Second Differentials in Arbitrary Feed-Forward Neural Networks," ThomsonCSF /SDC, 1995.

[116] Y.H. Tay, "Off-line Handwriting Recognition using artificial Neural Network and Hidden Markov Model," University of Nantes and University Technologi Malaisia, Thèse de doctorat 2002.

[117] Harold Mouchère, "Étude des mécanismes d'adaptation et de rejet pour l'optimisation de classifieurs : Application à la reconnaissance de l'écriture manuscrite en-ligne.," l'Institut National des Sciences Appliquées de Rennes (INSA), Thèse de doctorat 2007.

[118] M. Schenkel, I. Guyon, and D. Henderson, "Online cursive script recognition using time delay neural networks and hidden Markov models," *Machine vision and applications, Special Issue on Cursive Script Recognition*, vol. 8, pp. 215-223, 1995.

[119] Juan Manuel Torres-Moreno, "Apprentissage et généralisation par des réseaux de neurones: étude des nouveaux algorithmes constructifs," institut National Polytechnique de Grenoble, Thèse de doctorat 1997.

[120] D. Opitz and R. Maclin, "Popular Ensemble Methods: An Empirical Study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169-198, 1999.

[121] H. Schwenk and Y. Bengio, "AdaBoosting Neural Networks: Application to on-line Character Recognition," in *International Conference on Artificial Neural Networks*, 1997, pp. 967-972.

[122] Y. LeCun et al., , Touretzky and David, Eds.: (NIPS*89), Morgan Kaufman, 1990, ch. Handwritten digit recognition with a back-propagation network.

[123] J. Shawe-Taylor, "Introducing invariance: a principled approach to weight sharing," in *International Conference on Neural Network , IEEE World Congress on Computational Intelligence*, 1994, pp. 345-349.

[124] D. de Ridder, "Shared weights neural networks in image analysis," Delft University of Technology, Thèse de doctorat 1996.

[125] Hui Zhu, L. tang, and Peng Liu, "An mlp-orthogonal quassian mixture hybrid model for chinese bank check printed numeral recognition," *International Journal of Document Analysis and Recognition*, vol. 8, pp. 27-34, 2006.

[126] J.G. Wilpon, L.R. Rabiner, C.-H. Lee, and E.R. Goldman, "Automatic recognition of keywords in unconstrained speech using hidden markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 1870-1878, 1990.

[127] C.M. Bishop, *Neural Networks for Pattern Recognition*, 0, Ed.: Oxford University Press, 1995.

## Résumé

Les travaux présentés dans le cadre de cette thèse portent sur l'étude, la conception, le développement et le test d'un système de reconnaissance de structures manuscrites bidimensionnelles. Le système proposé se base sur une architecture globale qui considère le problème de reconnaissance en tant qu'optimisation simultanée de la segmentation, de la reconnaissance de symboles, et de l'interprétation. Le premier cadre d'applications a été celui d'un système de reconnaissance d'expressions mathématiques manuscrites. La difficulté du problème se situe aux trois niveaux évoqués. La segmentation est complexe du fait de la grande liberté de composition d'une expression, avec notamment la possibilité de symboles multi-traits non séquentiels ; la reconnaissance doit affronter un nombre élevé de classes et en particulier, gérer les situations de formes non-apprises ; l'interprétation peut-être ambiguë du fait du positionnement spatial approximatif. La solution proposée repose sur la minimisation d'une fonction de coût global qui met en compétition des coûts de reconnaissance et des coûts structurels pour explorer un vaste espace de solutions. Les résultats obtenus sont très compétitifs et prometteurs comparés à ceux de la littérature. Nous avons finalement montré la généricité de notre approche en l'adaptant à la reconnaissance d'un autre type de langage 2D, celui des représentations graphiques de type organigramme.

**Mots-clés :**
Reconnaissance de formes, écriture manuscrite, langages bidimensionnels, expressions mathématiques, analyse structurelle, analyse syntaxique, évaluation

## Abstract

### Recognition of two-dimensional structures: application on online handwritten mathematical expressions

This thesis focuses on the study, conception, development and testing of a recognition system for two-dimensional handwritten structures. The proposed system is based on a global architecture that considers the problem of recognition as simultaneous optimization of segmentation, symbol recognition, and interpretation. In this framework, we have first designed a system to recognize handwritten mathematical expression. All the three problems of segmentation, recognition and interpretation are not straightforward. Segmentation is complex because of the large freedom for composing an expression, since delayed multi-stroke symbols are considered. Recognition has to face a large number of classes, and to deal with the problem of unknown pattern, and interpretation suffers for the fuzzy nature of spatial relationships. We have defined a solution which minimizes a global cost function where recognition costs and structural costs are combined and a large exploration of the space of solutions is proposed. The results are very promising and competitive compared to those of the literature. We have finally shown the generality of our approach in adapting the system to the recognition of another 2D language, which is used to design handwritten flowcharts.

**Keywords:**
Pattern recognition, handwriting, bidimensional languages, mathematical expressions, structural analysis, syntactic analysis, benchmarking

Discipline :
Automatique et Informatique appliquée